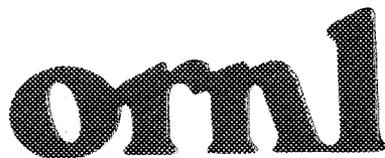




3 4456 0316022 8

ORNL/TM-11595



**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

**A Technical Description of
Enhancements to the Front-End User
Interface for the Worldwide Household
Goods Information System for
Transportation Modernization
(WHIST-MOD)**

J. P. Loftis
T. L. James
P. M. Spears

OAK RIDGE NATIONAL LABORATORY
CENTRAL RESEARCH LIBRARY
CIRCULATION SECTION
2200 FORD LN
LIBRARY LOAN COPY
DO NOT TRANSFER TO ANOTHER PERSON
If you wish someone else to see this
report, send its name with report and
the library will arrange a loan.

OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

NTIS price codes—Printed Copy: A05 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Energy Division

**A TECHNICAL DESCRIPTION OF
ENHANCEMENTS TO THE FRONT-END USER INTERFACE FOR THE
WORLDWIDE HOUSEHOLD GOODS INFORMATION SYSTEM FOR
TRANSPORTATION MODERNIZATION (WHIST-MOD)**

Jane P. Loftis
Teresa L. James*
P. Mark Spears

*University of Tennessee, Knoxville

Date Published: August 1990

Prepared for the
Military Traffic Management Command
Falls Church, Virginia 22041
under Interagency Agreement DOE No. 1405-1405-A2

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6027
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under
Contract No. DE-AC05-84OR21400

MARTIN MARIETTA ENERGY SYSTEMS LIBRARIES



3 4456 0316022 8

CONTENTS

	Page
LIST OF FIGURES	v
ACRONYMS AND TERMS	vii
ACKNOWLEDGMENTS	ix
ABSTRACT	xi
1. INTRODUCTION AND BACKGROUND	1
2. DESIGN CONSIDERATIONS	3
2.1 USER DEFINITION	3
2.1.1 Action Officers	3
2.1.2 Analysts	3
2.1.3 Managers	4
2.1.4 Computer Skills of Users	4
2.2 SYSTEM FLEXIBILITY	4
3. THE INTERFACE	7
3.1 BASIC DESIGN	7
3.2 ENHANCEMENTS TO THE INTERFACE	9
3.2.1 Need for Enhancements	9
3.2.2 Discussion of Enhancements	9
4. TECHNICAL DESCRIPTION OF ENHANCEMENTS	13
4.1 BACKGROUND	13
4.2 ENTER ENHANCEMENT	16
4.3 SORT ENHANCEMENT	18
4.4 SUBSTRING SEARCH ENHANCEMENT	20
5. SUMMARY	23
APPENDIX	A-1

LIST OF FIGURES

	Page
Fig. 3.1. Components and relationships of the user interface	8
Fig. 3.2. The basic design of the lower-level Carrier Selection Screen	10
Fig. 3.3. The design of the Carrier Selection Screen with the enhancements	11
Fig. 4.1. The technical design of the Carrier Selection Screen	14
Fig. 4.2. Definition of symbols used in the process flow diagrams	15
Fig. 4.3. Process flow diagram of the basic design	15
Fig. 4.4. Process flow diagram with the enter enhancement	17
Fig. 4.5. Process flow diagram with the sort enhancement	19
Fig. 4.6. Process flow diagram with the string search enhancement.	21

ACRONYMS AND TERMS

COS	code of service
MTMC	Military Traffic Management Command
MTPP	Directorate of Personal Property, MTMC
ORACLE	Relational Database Management System and related software
ORNL	Oak Ridge National Laboratory
RDBMS	Relational Database Management System
SAS	Statistical Analysis and Data Management Toolset
SCAC	Standard Carrier Alpha Code
SQL	Standard Query Language
WHIST-MOD	Worldwide Household Goods Information System for Transportation Modernization

ACKNOWLEDGMENTS

The authors would like to thank the staff at the Directorate of Personal Property of the Military Management Traffic Command for their input and support. Several staff members at Oak Ridge National Laboratory also made significant contributions. Key persons include Tai-Lun Chiang, who assisted in the coding of the user interface applications, and Vickie Ng. Finally, the authors would like to thank Robin Noe for her highly professional support in the preparation and publication of this document.

ABSTRACT

The Directorate of Personal Property of the Military Traffic Management Command (MTMC) asked Oak Ridge National Laboratory (ORNL) to design a decision support system, the Worldwide Household Goods Information System for Transportation Modernization. This decision support system will automate tasks and provide analysis tools for evaluating the Personal Property Program, predicting impacts to the program, and planning modifications to the program to meet the evolving needs of military service members and the transportation industry. The system designed by ORNL consists of three application modules: system dictionary applications, data acquisition and administration applications, and user applications. The user applications module is divided into two phases: the data selection front-end interface and the postprocessing back-end interface.

This paper describes the prototyped front-end interface using ORACLE SQL*Forms, part of the ORACLE Relational Database Management System (RDBMS) toolset. The focus of this paper is a discussion of the need for enhancements to the initial design of the interface and the coding techniques used to prototype the enhancements. These enhancements make the front-end interface more flexible and easier to use by giving users options for identifying data to be used by the back-end interface. This report is based on in-depth interviews of MTMC staff, prototype meetings with the users, and the research and design work conducted at ORNL.

1. INTRODUCTION AND BACKGROUND

The Worldwide Household Goods Information System for Transportation Modernization (WHIST-MOD) being designed and prototyped by Oak Ridge National Laboratory (ORNL) is a decision support system for the various organizations of the Military Traffic Management Command (MTMC) that establish and implement the Personal Property Movement and Storage Program. The decision support system will benefit the staff of the Personal Property Program in their tasks of program evaluation and policy setting. This system is designed to access a distributed database through a powerful set of information management tools.

The prototype system offers users, even those with minimal computer experience, easy access to a large selection of data elements and the ability to easily formulate complex queries. In addition, users may perform special studies and one-time-only *ad hoc* queries. WHIST-MOD will be a dynamic system that evolves to meet the changing needs of the Directorate of Personal Property MTMC (MTPP) staff.

User requirements for the WHIST-MOD system were identified during the analysis phase of the project. During this phase ORNL identified three modules for prototyping to aid decision support activities at MTPP. These modules include applications to provide a description of the database (system dictionary applications), applications to manage data (data acquisition and administration applications), and applications to retrieve and display data from the database (user applications).

The user applications module is divided into two prototype design and development phases: a front-end interface and a postprocessing back-end interface. The front-end interface was prototyped using ORACLE Relational Database Management System (RDBMS) and two ORACLE products, SQL*Forms and SQL*Plus.* The front-end interface includes screens that allow users to choose, retrieve, and store a subset of data. These data are then passed to the back-end interface. The back-end interface will be prototyped using SAS, a data management and statistical analysis toolset. The back-end interface will allow the user to specify report types and formats and to produce output based on the dataset passed to SAS from the front-end interface.

The purpose of this document is to describe the enhancements that were added to the front-end interface prototyped using ORACLE and to give the coding techniques that were used to prototype the enhancements. Section 2 gives a brief description of the requirements for the interface. Section 3 describes the basic design of the interface and the enhancements that were needed to make the interface easier to use. The coding techniques that were used to prototype the enhancements are described in Sect. 4.

*SQL—Standard Query Language.

2. DESIGN CONSIDERATIONS

Two major considerations during the design phase of the front-end interface were the various needs of the user community and the need for a flexible interface to the database. The following sections discuss the system requirements that affected the design of the front-end interface prototyped in ORACLE.

2.1 USER DEFINITION

One of the challenges in the design of the software was to prototype an interface that served the needs of a disparate group of users. Three categories of users were identified during the analysis phase of the project:

- action officers,
- analysts, and
- managers.

2.1.1 Action Officers

Action officers are responsible for overseeing and analyzing specific areas of the Personal Property Program. They may be assigned to one of three divisions at MTPP: Rate Acquisition, Quality Assurance and Operations, or Management Support. Each of these divisions analyzes different data and produces different reports. Action officers are assigned to specific tasks within each division. For example, in the Rates Division, one action officer is assigned to oversee the mobile home rate solicitation while another action officer is responsible for the domestic rates solicitation. One of the system requirements was to prototype an interface that would meet the needs of the different divisions as well as the needs of each action officer.

2.1.2 Analysts

Analysts are responsible for preparing reports and special studies for managers and for outside organizations such as Congress and the Armed Services. They also produce reports to support action officer needs. Their primary responsibilities are to research and analyze problems, investigate proposed policy changes, and provide computer support.

Because of the nature of the analyst's job, no standard interface could be built to anticipate every issue that might arise. However, this interface does support the analyst's needs by allowing him/her to rapidly build complex queries of the database for many commonly asked questions.

2.1.3 Managers

Managers need timely, accurate data in order to perform the following tasks:

- to identify specific problems in the Personal Property Program,
- to determine where changes are needed,
- to monitor the program, and
- to report to outside agencies.

The interface was designed to allow managers to have direct access to the data they need to perform these tasks.

2.1.4 Computer Skills of Users

In addition to supporting users with different tasks, another challenge in the design of the interface was to prototype a system to support users with different levels of computer skills. In each category of users the amount of computer experience ranges from people with no previous computer experience to skilled computer programmers. A further consideration was the high turnover rate of the staff at MTPP.

To accommodate the users the interface has the following characteristics.

- It is simple enough to be used by people with minimal computer skills.
- It includes extensive help for first-time or infrequent users.
- It requires no knowledge of the database structure.
- It requires no knowledge of the database language.
- It is flexible enough to meet the needs of experienced as well as inexperienced users.

2.2 SYSTEM FLEXIBILITY

ORNL designed the front-end user interface to be flexible and generic enough to encompass the data access needs of all the users. Users can identify data for a variety of reports from this single interface. During the analysis phase of the project, a group of standard applications was identified as the reports the system needed to generate. MTPP staff and managers identified the following reports as the ones used most frequently:

- Average Net Weight Shipped,
- Average Cost of Shipments,
- Change in Rate Levels,
- Number of Shipments into Storage in Transit,
- Tonnage,
- Number of Shipments,
- Total Actual Cost (plus claims),
- Score Summary Statistics,
- Missed Pickup,
- Loss/Damage,

- Other Tender of Service Violations, and
- Number of Quality Assurance Actions.

The front-end interface is designed to support the identification and selection of data for all of these report types. These data will be passed to the back-end interface, where they will be used in the reports.

3. THE INTERFACE

3.1 BASIC DESIGN

The front-end user interface is designed to allow users to choose the subset of data they want to appear on a report. Figure 3.1 illustrates the relationships between the components of the prototyped user interface. The users move through the interface screens selecting parameters for data they want to include on a report in the following order.

- They select an application category on the application category screen (e.g., Rates).
- They select an application (report) on a user application screen (e.g., Average Net Weight Shipped).
- They select parameter categories [i.e., carriers, codes of service (COSs), geographic areas, and time periods] on the Parameter Selection Menu Screen.
- They select specific data from a reference table in any or all of the lower-level parameter selection screens (i.e., COS Selection Screen, Time Interval Selection Screen, Carrier Selection Screen, and Area Selection Screen).

The lower-level parameter selection screens retrieve and display parameters according to the parameter category the user has chosen. For example, if the user has chosen "code of service" on the Parameter Category Screen, the lower-level parameter screen performs a query that retrieves and displays all appropriate COSs.

Users identify the subset of data they want to appear on a report by selecting any of the following:

1. specific carriers (e.g., ABC Moving Company);
2. specific origins and destinations (e.g., Virginia to California);
3. specific codes of service (e.g., code 1A); and
4. specific time periods (e.g., May 1, 1990 to May 31, 1990).

As users make selections on the lower-level parameter selection screens, their choices are stored in a database table called the Conditions Table. An operating system script builds a query that retrieves data from the shipment tables based on the user choices stored in the Conditions Table. For example, using the parameter choices given as illustrations in this section, the script would build a query that retrieves data for all shipments moved by ABC Moving Company from Virginia to California under code 1A during May 1990. The data retrieved from this query are written to a dataset file that is used by SAS to generate reports and charts.

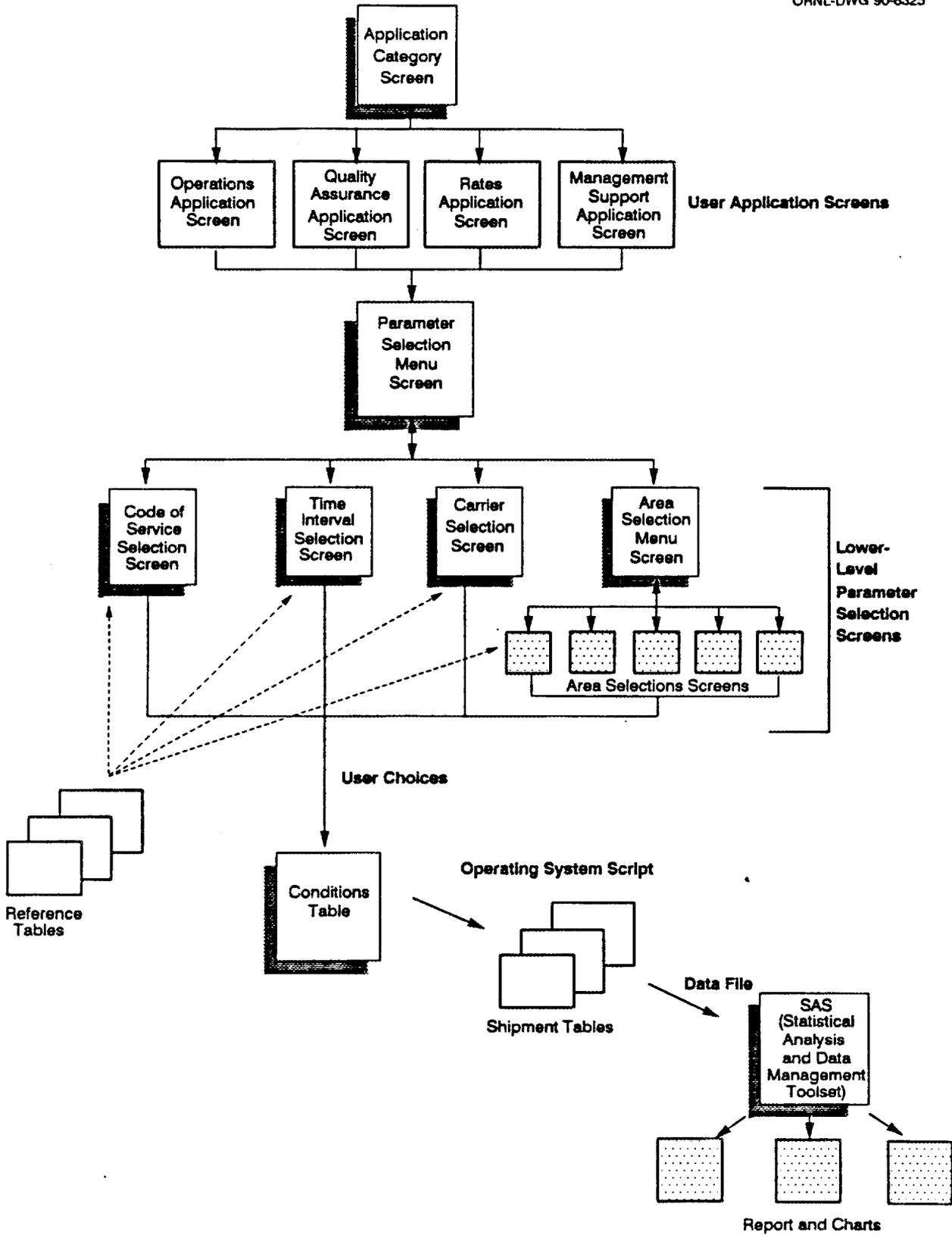


Fig. 3.1. Components and relationships of the user interface.

For more details on the design of the interface see Sect. 3 of the "User Interface in ORACLE for the Worldwide Household Goods Information System for Transportation Modernization (WHIST-MOD)" document."

3.2 ENHANCEMENTS TO THE INTERFACE

3.2.1 Need for Enhancements

The basic design of the interface worked well for the users and provided the flexibility needed to generate a variety of reports. However, because of the large volume of reference data, it was sometimes difficult for users to locate specific parameters.

To make the interface easier to use, ORNL provided enhancements to address the following problems:

- the retrieval of too many parameter choices and
- the ordering of the displayed parameters.

Because in its basic design the interface often retrieved and displayed more records than the users wanted, users needed more control over which records were displayed. For example, up to 700 carrier records could be displayed on the lower-level Carrier Selection Screen. Because only ten records were displayed on one page, users had to scroll through many pages to locate a specific carrier. This scrolling proved to be a tedious process for users who needed a report for a few specific carriers.

In addition to limiting the number of parameters displayed, users needed to be able to control the ordering of the displayed records. In the basic interface design, the Carrier, COS, and Area Lower-Level Parameter Screens displayed two columns—a code column and a description column (Fig. 3.2). The records displayed on these lower-level screens were sorted on the code column. Because some users are more familiar with carrier names than with Standard Carrier Alpha Codes (SCACs), users wanted the option of sorting the displayed records on the description column so they could search for carrier by name.

3.2.2 Discussion of Enhancements

Two enhancements were needed to address the problem of the large number of parameter choices retrieved. Initially an option was added to the lower-level parameter screens that allows users to enter specific codes or retrieve a complete list of codes. This "enter" enhancement, however, solves the problem of quickly specifying carriers only if users know the specific codes they need for a report. Users often know only the first letters of a code or description. Therefore, a second enhancement was added that allows users to search the database for records that begin with certain letters.

*T. James and J. Loftis, *User Interface in ORACLE for the Worldwide Household Goods Information System for Transportation Modernization (WHIST-MOD)*, ORNL/TM-11596, Martin Marietta Energy Systems, Inc., Oak Ridge Natl. Lab., July 1990.

User Name	AVERAGE NET WEIGHT SHIPPED	DOM
	SCAC	Carrier Name
	* AAAA	CLARK TRANSFER & STORAGE CO
	BBBB	CROWN MOVING & STORAGE INC OF ILLINOIS
	* CVLC	CARTWRIGHT VAN LINES INC
	* CVLS	CENTURY MOVING AND STORAGE
	CVLW	CONTINENTAL VAN LINES INC
	CVMO	CENTRAL VALLEY MOVING AND STORAGE
	CVMQ	COR-O-VAN MOVING & STORAGE CO
	CVNI	CENTRAL VAN LINES INC
	CVNS	CARDINAL VAN AND STORAGE
	CWMO	CROWN MOVING & STORAGE INC
Press CTRL F6 to commit SCACs.		

Fig. 3.2. The basic design of the lower-level Carrier Selection Screen. The code column contains the Standard Alpha Carrier Code (SCAC) and the description column contains the name of the carrier.

A further enhancement to the lower-level parameter screens allows users to choose whether they want displayed parameters to be sorted by the code field or by the description field. This "sort" enhancement allows users to see the retrieved data ordered in a familiar way.

Figure 3.3 shows the new design of the Carrier Selection Screen. This design incorporates the entering, sorting, and substring search enhancements.

User Name	AVERAGE NET WEIGHT SHIPPED	DOM
List or Enter Carrier Codes: L Sort by Carrier Code (C) or Description (D) : C String for Limiting Search : C		
SCAC	Carrier Name	
• CVLC	CARTWRIGHT VAN LINES INC	
• CVLS	CENTURY MOVING AND STORAGE	
CVLW	CONTINENTAL VAN LINES INC	
CVMO	CENTRAL VALLEY MOVING AND STORAGE	
CVMQ	COR-O-VAN MOVING & STORAGE CO	
CVNI	CENTRAL VAN LINES INC	
CVNS	CARDINAL VAN AND STORAGE	
CWMO	CROWN MOVING & STORAGE INC	
Press CTRL F6 to commit SCACs.		

Fig. 3.3. The design of the Carrier Selection Screen with the enhancements. This screen displays carrier parameters, ordered by Standard Alpha Carrier Codes (SCACs), for SCACs that begin with "C."

4. TECHNICAL DESCRIPTION OF ENHANCEMENTS

This section describes the technical aspects of the basic design and gives step-by-step instructions for implementing the enhancements discussed in Sect. 3.2.2. The following discussion begins with a technical description of the basic design, describing each enhancement one-by-one until the complete design with enhancements is presented. The Carrier Selection Screen is used throughout this section to provide examples.

The discussions that follow assume a basic familiarity with ORACLE SQL*Forms and SQL*Plus. To facilitate the discussions of the enhancements, only the components of the basic design that are relative to the enhancements are discussed. The complete code for the Carrier Selection Screen is included in the Appendix.

Field and user-defined triggers are referred to by their actual names in the code for the Carrier Selection Screen. To make the code easier to follow, all field names are italicized and all user-defined triggers are in bold type.

4.1 BACKGROUND

This section describes the technical design of the Carrier Selection Screen (Fig. 4.1). Initially there were three components in the design of this screen:

- a header block,
- a displayed base table/view block, and
- a hidden base table block.

The header block is a display-only control block. It consists of three fields:

- the user name,
- the name of the report, and
- the service category.

The *user_name* field contains the ORACLE user name, and the *app_name* field contains the name of the report selected by the user in a menu screen. The *service_category* field contains control information that is not relevant to this discussion.

The displayed base table/view block, block one, has the Carrier Approval View as its base table. It serves two purposes. It displays multiple (up to ten) carrier names and SCACs at a time, and it allows users to select particular carriers for inclusion on a report. Using symbols described in Fig. 4.2, the process flow of this form is illustrated in Fig. 4.3.

Block one contains three fields:

- *star*,
- *carrier_code*, and
- *carrier_name*.

The *star* field is an enterable, non-base-table field that indicates user selections. The form executes a trigger, **put_star**, which inserts an asterisk in the *star* field to indicate the

Header Block

User Name	AVERAGE NET WEIGHT SHIPPED	DOM
SCAC	Carrier Name	
AAAA	CLARK TRANSFER & STORAGE CO	
BBBB	CROWN MOVING & STORAGE INC OF ILLINO	
CVLC	CARTWRIGHT VAN LINES INC	
CVLS	CENTURY MOVING AND STORAGE	
CVLW	CONTINENTAL VAN LINES INC	
CVMO	CENTRAL VALLEY MOVING AND STORAGE	
CVMQ	COR-O-VAN MOVING & STORAGE CO	
CVNI	CENTRAL VAN LINES INC	
CVNS	CARDINAL VAN AND STORAGE	
CWMO	CROWN MOVING & STORAGE INC	
Press CTRL F6 to commit SCACs.		

Base Table/View Block

Fig. 4.1. The technical design of the Carrier Selection Screen.

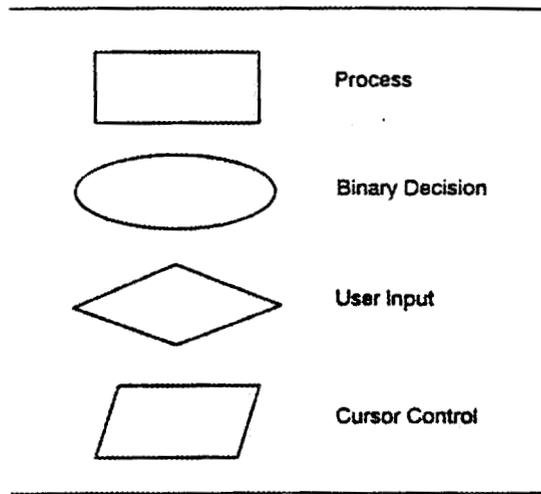


Figure 4.2. Definition of symbols used in the process flow diagrams.

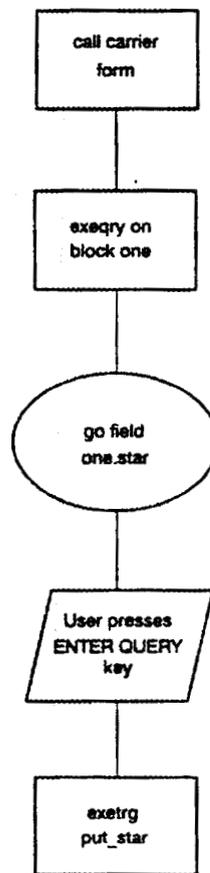


Fig. 4.3. Process flow diagram of the basic design.

user has selected a particular carrier or deletes the asterisk to indicate the user has canceled a previous selection. The *carrier_code* and *carrier_name* fields are base table fields that display the carrier data retrieved from the query of Carrier Approval View.

The hidden base table block, block two, has the Conditions Table as its base table. Block two is used to insert carrier parameters selected by the user into this table. The details about the functionality of block two are not relevant to this discussion of enhancements to the basic design.

4.2 ENTER ENHANCEMENT

The enter enhancement was added to the basic design to allow users to enter parameter codes as an alternative to scrolling through pages of data (see Sect. 3.7.2). Figure 4.4 illustrates the process flow of the design with the addition of the enter enhancement. In this diagram the *lentqry* trigger replaces the functionality of the *put_star* trigger in the basic design. This section provides the steps needed to add this enhancement.

- Step 1 Build a control block (block zero) that contains a *list_or_enter* field. This is a mandatory, enterable, one-character field that must contain an "L" or an "E."
- Step 2 Create a field-level trigger on the NEXT FIELD key that branches based on the value in the *list_or_enter* field. If the value is "L," go to block one and execute a query on the Carrier Approval View. If the value is "E" execute the *enter_list* trigger.
- Step 3 Create an *enter_list* trigger that checks the Conditions Table to see if the user has made previous selections. The details of checking for previous selections are not relevant to this discussion. The trigger also moves the cursor to the *carrier_code* field in block one, displays a help message, and waits for the user to enter a SCAC and press the ENTER QUERY key.
- Step 4 Create a block-level trigger on the ENTER QUERY key that branches depending on the value in the *list_or_enter* field. If the value is "L," call the *lentqry* trigger. If the value is "E," call the *centqry* trigger.
- Step 5 Create a block-level *lentqry* trigger to check whether the user is selecting or canceling a selection. The trigger enters an asterisk in the *star* field if the field is empty or deletes the asterisk if one exists.
- Step 6 Create a block-level *centqry* trigger that retrieves and displays the carrier name that corresponds to the entered carrier code and inserts an asterisk in the *star* field or deletes the displayed record.

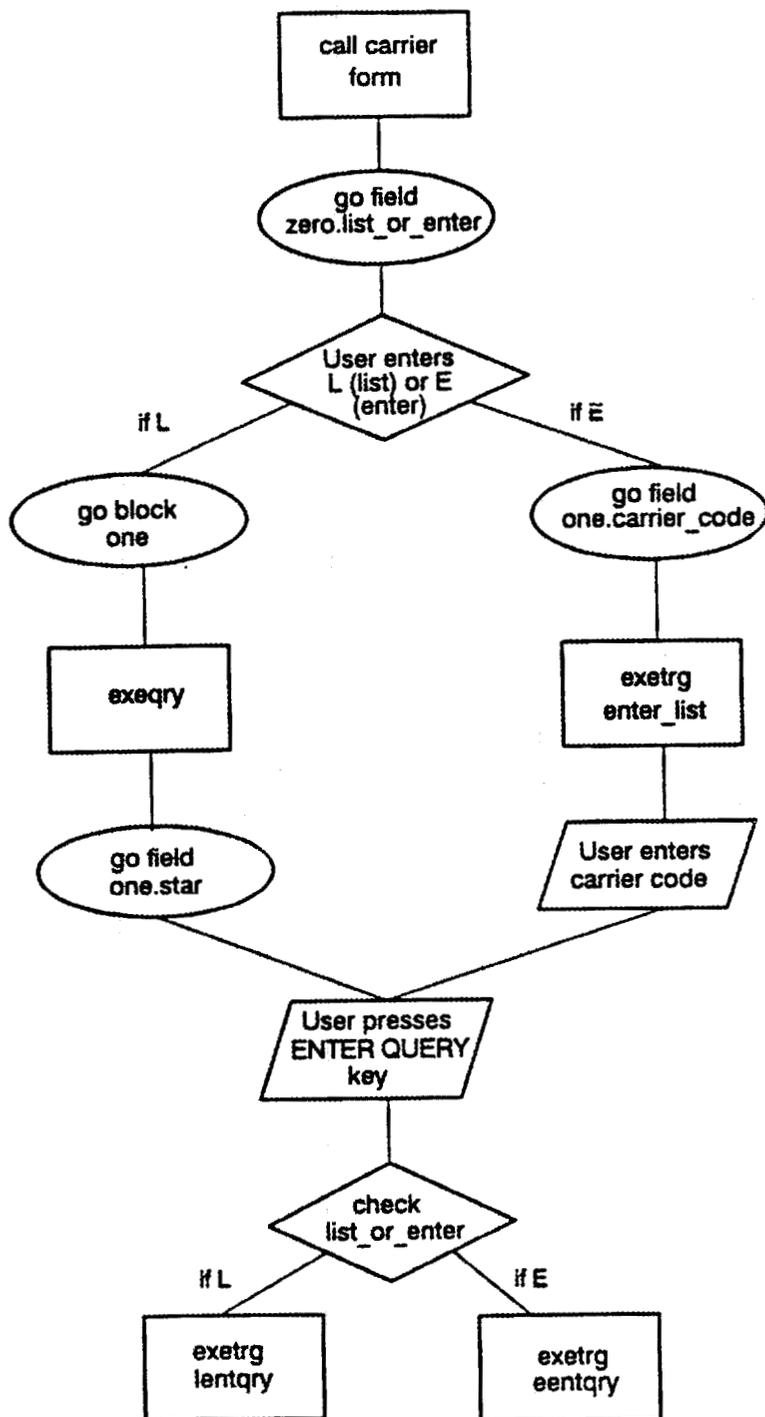


Fig. 4.4. Process flow diagram with the enter enhancement.

Once these new fields and triggers have been added to the basic design, users have two ways to choose carriers to subset the data to be included on a report. They may choose the "L" option, which executes a query on the Carrier Approval View, block one, and displays all appropriate carrier codes and names (providing the functionality of the basic design). They may also choose the "E" option and go directly to block one and enter the desired carrier codes.

4.3 SORT ENHANCEMENT

The sort enhancement was added to the basic design to allow users to choose the column for sorting displayed data parameters. Figure 4.5 illustrates the process flow with the addition of the sort enhancement. This section provides the steps needed to add this enhancement. The sort enhancement uses the control block built for the enter enhancement, but its functionality is used only if the user chooses the list option in the *list_or_enter* field.

- Step 1 Create a one-character field, *sort*, in block zero below the *list_or_enter* field.
- Step 2 Add control to the *list_or_enter* field so that if the user chose "L" in the *list_or_enter* field, the cursor moves to the *sort* field. The user enters "C" to sort by code or "D" to sort by description.
- Step 3 Create a nondisplayed field, *sortby*, that will contain the string "CARRIER_CODE" if the user has entered "C" or "CARRIER_NAME" if the user has entered "D".
- Step 4 Create a *check_sort* trigger that branches to call *put_scac* if the user has entered "C" or calls *put_carrier_name* if the user has entered "D".
- Step 5 Create a *put_scac* trigger that puts the string "CARRIER_CODE" into the *sortby* field.
- Step 6 Create a *put_carrier_name* trigger that puts the string "CARRIER_NAME" into the *sortby* field.

The remaining steps discuss an undocumented feature in ORACLE that allows users to choose the column on which the displayed parameters are sorted. This functionality is easy to implement in SQL*Plus using an ampersand, as the following example illustrates.

```
SELECT columns FROM table ORDER BY &sortby
```

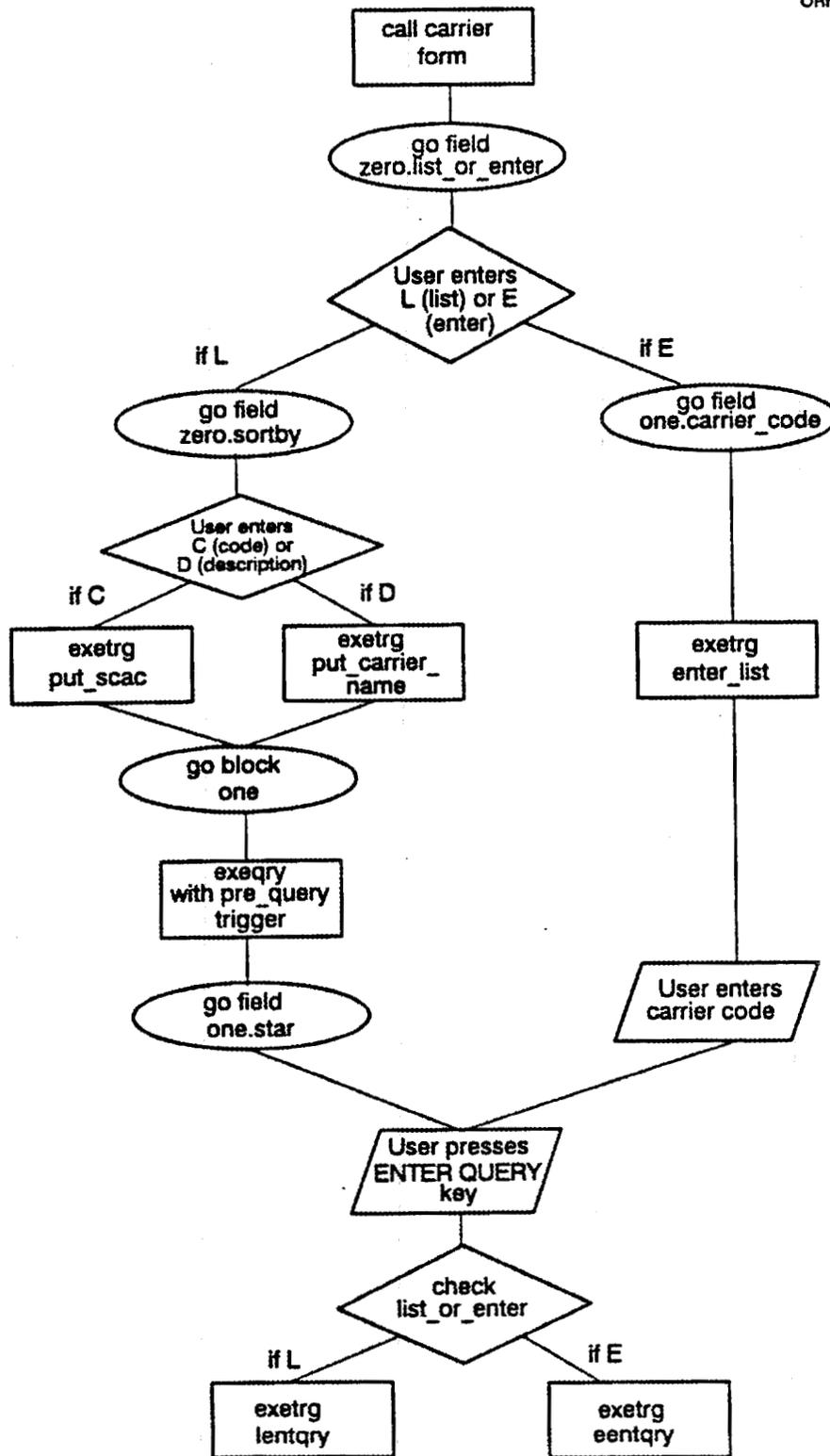


Fig. 4.5. Process flow diagram with the sort enhancement.

However, SQL*Forms does not support the use of an ampersand in an ORDER BY clause. To provide this functionality in SQL*Forms, the programmer must create a SELECT/ORDER BY statement in a PRE-QUERY trigger. The syntax for this statement follows.

PRE-QUERY:

```
SELECT '#LIKE "' || = base_block.last_field || '%"  
ORDER BY ' || :control_block.order_by_field  
INTO :base_block.last_field FROM dual
```

When this PRE-QUERY trigger is fired, it places the LIKE/ORDER BY string into the last field of the base table block and appends the LIKE/ORDER BY string to the base table query. This field in the base table block must be long enough to contain the string. Depending on the names of the blocks and fields used, the required length of the last field will vary. Steps 7 and 8 discuss implementing this feature using the *sortby* field created in the previous steps. In the example, *carrier_name* is the last field in the base table block on the Carrier Selection Screen.

Step 7 Create a PRE-QUERY trigger in block one.

PRE-QUERY:

```
SELECT '#LIKE "' || :one.carrier_name || ' %"  
ORDER BY ' || :zero.sortby INTO one.carrier_name FROM dual
```

Step 8 Increase the size of the *carrier_name* field, the last field in block one, to a length of 48 characters so it is long enough to contain the LIKE/ORDER BY string.

These new fields and triggers enhance the design for users who want to query the database for a list of appropriate carrier codes and names. They may now choose how the displayed data will be ordered by entering a "C" or a "D" in the *sort* field.

4.4 SUBSTRING SEARCH ENHANCEMENT

The substring search enhancement was added to the basic design to allow users to enter up to two characters to search the database for records that begin with certain letters. (This feature could allow users to search for any number of beginning, ending, or middle characters; however, WHIST-MOD users needed only the initial two-character search capability.) Figure 4.6 illustrates the design process flow with the addition of the substring search enhancement. This section provides the steps needed to add this enhancement. The substring search enhancement uses the control block built for the enter enhancement, but its functionality is used only if the user chooses the list option on the *list_or_enter* field.

Step 1 Create a two-character enterable field, *string*, in block zero beneath the *sort* field.

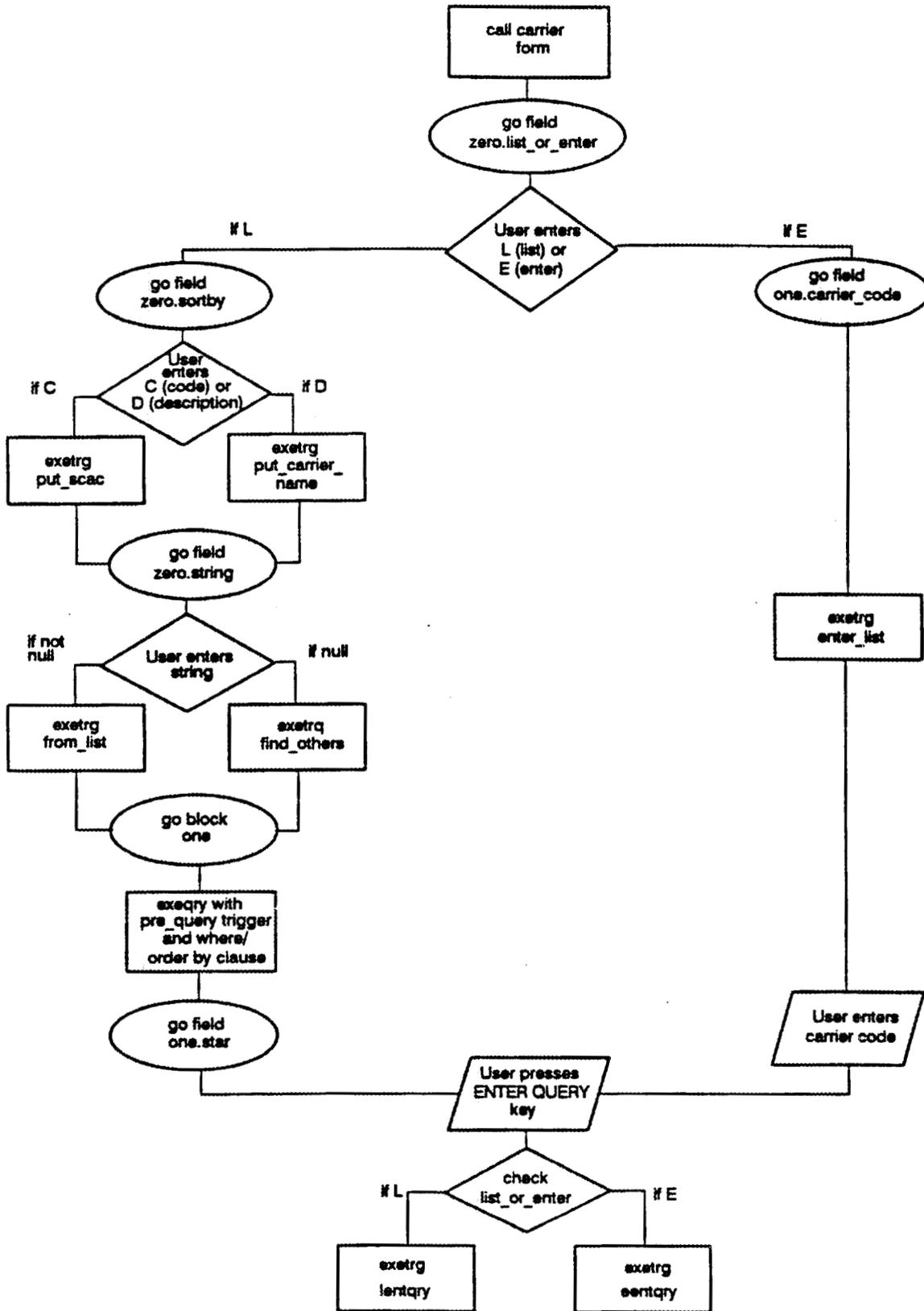


Fig. 4.6. Process flow diagram with the string search enhancement.

- Step 2** Create a field-level trigger on the NEXT FIELD key that branches based on the value in the *string* field. (This branching is optional, depending on the purpose of the form.) If the value is null, execute the `find_others` trigger. If the value is not null, execute the `from_list` trigger.
- Step 3** Create a `find_others` trigger that checks the Conditions Table to see if the user has made previous selections. The details of checking for previous selections are not relevant to this discussion. This trigger also executes a query on block one and positions the cursor in the *star* field.
- Step 4** Create a `from_list` trigger that deletes previous selections from the Conditions Table that do not match the string entered by the user. The details of this deletion process are not relevant to this discussion. This trigger also executes a query on block one and positions the cursor in the *star* field.
- Step 5** Modify the default WHERE/ORDER BY clause in block one, the base table block, to include the following code.

```

WHERE
(:zero.sortby = 'CARRIER_CODE' and
carrier_code like :zero.string || '%')
or
(:zero.sortby = 'CARRIER_NAME' and
carrier_name like :zero.string || '%')
or
(:zero.string is null and
zero.sortby like '%')

```

The fields and triggers added for the substring search enhancement are used in the default WHERE clause to retrieve data from the Carrier Approval View when a query is executed on this block. The substring search and the sort enhancement work in conjunction with each other. If the user enters "C" in the *sort* field, the WHERE clause restricts the query to those carrier CODES that begin with the value in the *string* field. If the user enters "D" in the *sort* field, the WHERE clause restricts the query to those carrier NAMES that begin with the value in the *string* field. For example, if the user chooses "C" in the *sort* field and "D" in the *string* field, the query will return all carriers whose carrier code begins with "D." If the user chooses not to enter a substring for searching, the query retrieves all carrier names and codes.

5. SUMMARY

The staff at MTPP are responsible for monitoring almost one million shipments annually. Based on the shipment data, they must evaluate the Personal Property Program and investigate the effects of policy changes. Currently, MTPP staff spend the majority of their time manually gathering, organizing, checking, and distributing data; supplying information to other federal agencies; and interacting with carriers concerning data problems. They have little time to analyze data that identify trends, predict impacts of trends and other changes on the Personal Property Program, or to recommend policy changes based on their analysis of the program.

The user application module designed and prototyped for WHIST-MOD will automate many of the tasks now performed manually by MTPP staff. This automation will allow the staff to spend more time analyzing the Personal Property Program. In addition, the front- and back-end interfaces allow the user quick access to data and the ability to manipulate and display these data easily.

This document describes three enhancements to the front-end interface already prototyped at ORNL—the enter, sort, and substring search enhancements. These enhancements greatly improved the ease of use and the flexibility of the WHIST-MOD interface. They could readily be adapted, singly or in combination, to other interfaces on projects using SQL-based RDBMSs.

APPENDIX


```

; Generated by SQL*Forms Version 2.0.21 on Thu Mar 1 11:41:56 1990
; Application owner is TAILUN. Application name is scac_par
; (Application ID is 0)
; -----
; Generate by SQL*Forms Version 2.0.21 on Thu Feb 8 14:10:09 1990
; The Carrier Selection Screen performs the following functions:
; 1. Allows the user to choose the SCACs for an output report by
; a. Entering the SCACs or
; b. Selecting the SCACs available from a list of applicable
; SCACs.
; 2. Checks to ensure the SCAC(s) chosen by the user (whether ;
; entered or selected) apply to the service category and COS(s) ;
; chosen.
; 3. Stores historical SCAC(s) in the conditions table for any
; carrier, when appropriate.
;The SCAC screen is called by the main_parameter screen when the user
;chooses the SCAC option.
;The SCAC(s) chosen by the user is written to the user-owned condition
;table.
;When the user exits the application, she/he returns to the
;main_parameter screen.
;Application Title :
CARRIER PARAMETER SELECTION SCREEN
;ORACLE workspace size :

;Block name / Description :
**key-exit
;SQL>
;Enables the EXIT function.
#exemacro exit;

;Message if value not found :

;Must value exist Y/N :
Y
;Block name / Description :
**KEY-NXTSET
;SQL>
;Enables the NXTSET function.
#exemacro nxtset;

;Message if value not found :

;Must value exist Y/N :
Y
;Block name / Description :

```

```

**key-others
;SQL>
;Turns off all ORACLE function keys that are not enabled in the
;application.
#exemacro null;

;Message if value not found :
*That function is disabled in this portion of the application.
;Must value exist Y/N :
Y
;Block name / Description :
**KEY-PRINT
;SQL>
;Enables the PRINT function.
#exemacro print;

;Message if value not found :

;Must value exist Y/N :
Y
;Copies user identification information (put in the global area from the
;main parameter application) into the header block and
;moves the cursor to the select_or_enter field in block zero.
;Block name / Description :
**key-startup
;SQL>
#copy global.g_user_name :header.user_name
/
;Message if value not found :

;Must value exist Y/N :
Y
#copy global.g_app_name :header.app_name
/
;Message if value not found :

;Must value exist Y/N :
Y
#copy global.g_analysis_set_num :header.analysis_set_num
/
;Message if value not found :

;Must value exist Y/N :
Y
#copy global.g_service_category :header.service_category
/
;Message if value not found :

```

```

;Must value exist Y/N :
Y
#copy '0' :zero.rec_count
/
;Message if value not found :

;Must value exist Y/N :
Y
#copy :header.service_category :zero.category
/
;Message if value not found :

;Must value exist Y/N :
Y
#exemacro goblk zero; gofld select_or_enter; help;

;Message if value not found :

;Must value exist Y/N :
Y
;Block two has the conditions table as the base table. All inserts or
;deletions in this screen are performed to this table.
;Block name / Description :
*two/two
;Enter default WHERE and ORDER BY clause :
where user_id = user and terminal_id = userenv('TERMINAL') and
      analysis_seq_num = :header.analysis_set_num and
      carrier_code is not null

;Table name :
CONDITIONS
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1 / 3
;This trigger is used to position the cursor one record past the last
;record returned from a qualified query of the conditions table.
;Field name :
*bottom2
;SQL>
;Until last record ...
$first
select 'x' from system.dual where :two.user_id is not null
/
;Message if value not found :
$continue $stop
;Must value exist Y/N :

```

```

N
;Go to next record.
$continue
#exemacro exetry two_nxtrec;
/
;Message if value not found :
$first $first
;Must value exist Y/N :
N
;You are now postioned past the last record.
$stop
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;The del_recs trigger is called when the user has entered a substring
;search. This trigger deletes all records that have been previously
;entered by this user during this session.
;Field name :
*del_recs
;SQL>
;Delete records until no records exist.
select 'x' from dual where :two.user_id is not null
/
;Message if value not found :
$continue $exit
;Must value exist Y/N :
N
;Begin loop
$continue
select 'x' from dual where :two.user_id is not null
/
;Message if value not found :
$continue2 $commit
;Must value exist Y/N :
N
;Delete record.
$continue2
#exemacro delrec;
/
;Message if value not found :
$continue $continue
;Must value exist Y/N :
N
;Commit deletion of all records.

```

```

$commit
#exemacro commit;
/
;Message if value not found :
$exit $exit
;Must value exist Y/N :
N
$exit
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;The search_and_destroy trigger is called from the sentqry and the
;eentqry triggers. Its purpose is to delete a record when the user is
;cancelling a selection.
;Field name :
*search_and_destroy
;SQL>
;If the record in block one matches the record in block two or if the
;record is blank, go to stop, else check next record.
$first
select 'x' from system.dual where :one.carrier_code = :two.carrier_code
OR
    :two.user_id is null
/
;Message if value not found :
$stop $continue
;Must value exist Y/N :
N
;Check next record.
$continue
#exemacro exetry two_nxtrec;
/
;Message if value not found :
$first $first
;Must value exist Y/N :
N
;If the record is blank do nothing; otherwise, delete the record.
$stop
#exemacro case :two.user_id is
    when '' then null;
    when others then exetry wipe;
end case;

;Message if value not found :

```

```

;Must value exist Y/N :
Y
;The purpose of the top2 trigger is to position the cursor at the first
;record in block two, so that any loop from here considers every record
;in that block.
;Field name :
*top2
;SQL>
;Until rec_count = 0 (which means cursor is pointing at first record).
$first
select 'x' from system.dual where :zero.rec_count != 0
/
;Message if value not found :
$continue $stop
;Must value exist Y/N :
N
;Decrement the counter and back up one record.
$continue
#exemacro exetrg two_prvrec;
/
;Message if value not found :
$first $first
;Must value exist Y/N :
N
$stop
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;The two_nxtrec trigger moves the cursor to the next record and
;increments the record counter, which keeps track of the current record.
;Field name :
*two_nxtrec
;SQL>
select 'x' from system.dual where :two.user_id is not null
/
;Message if value not found :
$continue $end
;Must value exist Y/N :
N
$continue
select :zero.rec_count + 1 into :zero.rec_count from system.dual
/
;Message if value not found :

```

```

;Must value exist Y/N :
N
#exemacro nxtrec;
/
;Message if value not found :
$end $end
;Must value exist Y/N :
N
$end
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;The two_prvrec decrements the record counter, which keeps track of the
;current record, and moves the cursor back one record.
;Field name :
*two_prvrec
;SQL>
select 'x' from system.dual where :zero.rec_count != 0
/
;Message if value not found :
$continue $end
;Must value exist Y/N :
N
$continue
select :zero.rec_count - 1 into :zero.rec_count from system.dual
/
;Message if value not found :

;Must value exist Y/N :
N
#exemacro prvrec;
/
;Message if value not found :
$end $end
;Must value exist Y/N :
N
$end
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;The wipe trigger deletes nonblank records in block two.

```

```
;Field name :
*wipe
;SQL>
#exemacro case :two.rowid is
  when '' then clrrec;
  when others then delrec;
end case;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
dummy
;Type of field :
CHAR
;Length of field / Display length / Query length :
1 / 1 / 1
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
2
;Column :
60
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
```

```
N
;Help message :

;Lowest value :

;Highest value :

;Field name :
TERMINAL_ID
;Type of field :
CHAR
;Length of field / Display length / Query length :
8 / 8 / 8
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
2
;Line :
7
;Column :
1
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
Y
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
Enter value for : TERMINAL_ID
```

```
;Lowest value :  
  
;Highest value :  
  
;Field name :  
CARRIER_CODE  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
4 / 4 / 4  
;Is this field in the base table Y/N :  
Y  
;Is this field part of the primary key Y/N :  
N  
;Default value :  
  
;Page :  
2  
;Line :  
13  
;Column :  
1  
;Prompt :  
  
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
Y  
;Hide value of field Y/N :  
N  
;SQL>  
  
;Is field mandatory Y/N :  
N  
;Is field fixed length Y/N :  
N  
;Auto jump to next field Y/N :  
N  
;Convert field to upper case Y/N :  
N  
;Help message :  
Enter value for : CARRIER_CODE  
;Lowest value :  
  
;Highest value :
```

```
;Field name :
ANALYSIS_SEQ_NUM
;Type of field :
NUMBER
;Length of field / Display length / Query length :
44 / 10 / 44
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
2
;Line :
7
;Column :
14
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
Y
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
Enter value for : ANALYSIS_SEQ_NUM
;Lowest value :

;Highest value :

;Field name :
USER_ID
```

```
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
30 / 30 / 30  
;Is this field in the base table Y/N :  
Y  
;Is this field part of the primary key Y/N :  
N  
;Default value :  
  
;Page :  
2  
;Line :  
7  
;Column :  
35  
;Prompt :  
  
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
Y  
;Hide value of field Y/N :  
N  
;SQL>  
  
;Is field mandatory Y/N :  
Y  
;Is field fixed length Y/N :  
N  
;Auto jump to next field Y/N :  
N  
;Convert field to upper case Y/N :  
N  
;Help message :  
Enter value for : USER_ID  
;Lowest value :  
  
;Highest value :  
  
;Field name :  
carrier_name  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :
```

```

48 / 48 / 48
;Is this field in the base table Y/N :
N
;Default value :

;Page :
2
;Line :
10
;Column :
15
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :

;Field name :

;Block one performs the following functions;
; If the user has chosen the LIST (L) option
;   1. It displays all applicable SCAC(s) - according to service ;
category selected and COS(s), if they have been chosen.
;   2. It sorts the display by the field (SCAC or Carrier Name) chosen
;      by the user in block zero.
;   3. It displays only those SCACs that fall within a substring ;
search, if the user entered this field in block one.

```

```

;      4. It puts an * beside the each choice the user has selected and
;      copies the record selected to block two, a hidden block.
; If the user has chosen the ENTER (E) option
;      1. It displays any SCACs which have been selected by this user ;
;      during the current session.
;      2. It moves the cursor to the carrier code field so that the user
;      may enter the first SCAC.
;      3. It places an * beside the entry the user selects, validates the
;      SCAC by checking the COS(s) and/or service category if ;
;      appropriate, retrieves historical SCACs for that carrier, if ;
;      appropriate, and copies the record to block two, a hidden ;      block.
;When the user commits a transaction, she/he commits all records that
;have been copied to block two into the user-owned conditions table.
;(The user may cancel a selection, which deletes the record in block two
;and in the conditions table, if a commit is performed.)
;Block name / Description :
*one/one
;Enter default WHERE and ORDER BY clause :
where carrier_code in (select carrier_code from approved_carrier
where code_of_service_approved in (select code_of_service from
conditions
where user_id = user and terminal_id = userenv('TERMINAL') and
analysis_seq_num = :header.analysis_set_num)
OR (0 = (select count(code_of_service) from conditions where
user_id = user
and terminal_id = userenv('TERMINAL') and
analysis_seq_num = :header.analysis_set_num)
AND (:zero.category in ('ALL','OTHER','VM') or
(:zero.category = 'INTL' and code_of_service_approved in
('4','5','6','7','8','J','T')) or
(:zero.category = 'MH' and code_of_service_approved = 'S') or
(:zero.category = 'INTRA' and
code_of_service_approved in ('1B','2B'))
or (:zero.category = 'INTER' and
code_of_service_approved in ('1A','2A')) or
(:zero.category = 'DOM' and
code_of_service_approved in ('1A','1B','2A','2B')) or
(:zero.category = 'DPM' and
code_of_service_approved like 'B_' or code_of_service_approved
like 'H_'))))
AND ((:zero.sortby = 'CARRIER_CODE' and
carrier_code like :zero.string||'%' )
or (:zero.sortby = 'CARRIER_NAME' and
carrier_name like :zero.string||'%' )
or (:zero.string is null and :zero.sortby like '%'))

;Table name :
```

```

APPROVED_CARRIER_VIEW
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
10 / 10
;Base crt line ?
11
;How many physical lines per record ?
1
;This trigger allows for the dynamic ORDER BY clause on the query of the
;approved_carrier view.
;It places the field chosen by the user and contained in the sortby
;field into block zero as the argument to the order by clause.
;Field name :
*PRE-QUERY
;SQL>
select '#like '''||:one.carrier_name||'%' order by '||:zero.sortby
into one.carrier_name from dual

;Message if value not found :

;Must value exist Y/N :
Y
;The Post-Query trigger places an * beside any previous SCAC selections
;made by this user during this session.
;Field name :
*POST-QUERY
;SQL>
select '*' into :one.star from system.dual where :one.carrier_code in
(select carrier_code from conditions where
user_id = user AND terminal_id = userenv('TERMINAL') AND
analysis_seq_num = 1 AND carrier_code is not null)

;Message if value not found :

;Must value exist Y/N :
N
;Allows the user to cancel what they have chosen and start over. This
;trigger does the following.
; 1. It clears records in block two, a hidden block.
; 2. If the user is listing SCACs, it performs another query of
; approved_carrier_view is performed.
;Field name :
*KEY-CLRBLK
;SQL>
#exemacro goblk two; clrblk; goblk one;
/

```

```
;Message if value not found :
```

```
;Must value exist Y/N :
```

```
N
```

```
#exemacro case :zero.select_or_enter is
  when 'L' then exeqry; gofld :one.star;
  when 'E' then clrblk; gofld :one.carrier_code;
  when others then exetrg se_error;
end case;
```

```
;Message if value not found :
```

```
;Must value exist Y/N :
```

```
Y
```

```
;Calls the scommit trigger if the user has selected SCAC(s) from a list
;or calls the ecommit if the user has entered SCAC(s).
```

```
;Field name :
```

```
*key-commit
```

```
;SQL>
```

```
#exemacro case :zero.select_or_enter is
  when 'L' then exetrg scommit;
  when 'E' then exetrg ecommit;
  when others then exetrg se_error;
end case;
```

```
;Message if value not found :
```

```
;Must value exist Y/N :
```

```
Y
```

```
;Calls the sentqry trigger if the user is listing SCAC(s) or
;calls the eentqry trigger if the user is entering SCAC(s).
```

```
;Field name :
```

```
*key-entqry
```

```
;SQL>
```

```
#exemacro case :zero.select_or_enter is
  when 'L' then exetrg sentqry;
  when 'E' then exetrg eentqry;
  when others then exetrg se_error;
end case;
```

```
;Message if value not found :
```

```
;Must value exist Y/N :
```

```
N
```

```
;Enables the NXTREC function.
```

```
;Field name :
```

```
*key-nxtrec
```

```

;SQL>
#exemacro nxtrec;

;Message if value not found :

;Must value exist Y/N :
Y
;Enables the PRVREC function.
;Field name :
*key-prvrec
;SQL>
#exemacro prvrec;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
*del_old_scac
;SQL>
;This trigger deletes the historic SCACs from the conditions table that
;do not correspond to the carriers that are currently chosen.
$step1
#exemacro goblk three; exeqry;
/
;Message if value not found :

;Must value exist Y/N :
N
;This step determines whether there are historic SCAC(s) corresponding
;to the carrier(s) the user has chosen.
$step2
select 'x' from system.dual where
:three.old_carrier_code is null
/
;Message if value not found :
$end $step3
;Must value exist Y/N :
N
;This step goes to the first record in block two.
$step3
#exemacro goblk two; exetry top2;
/
;Message if value not found :

;Must value exist Y/N :
N

```

```

;Looks for a record in block two.
$step3b
select 'x' from system.dual where
:two.user_id is null
/
;Message if value not found :
$send $step4
;Must value exist Y/N :
N
;Finds the record in block two that matches the current historic SCAC
;in block three.
$step4
select 'x' from system.dual where
:three.old_carrier_code = :two.carrier_code
/
;Message if value not found :
$step5 $step6
;Must value exist Y/N :
N
;Delete the record in block two.
$step5
#exemacro delrec;
/
;Message if value not found :
$step7 $step7
;Must value exist Y/N :
N
;Increment the counter.
$step6
#exemacro exetrg two_nxtrec;
/
;Message if value not found :
$step3b $step3b
;Must value exist Y/N :
Y
$step7
#exemacro goblk three; nxtrec;
/
;Message if value not found :
$step2 $step2
;Must value exist Y/N :
N
$send
#exemacro null;

;Message if value not found :

```

```

;Must value exist Y/N :
Y
;This trigger applies ONLY to SCAC(s) that have been entered by the
;user, not to those that have been selected from a list.
;Field name :
*ecommit
;SQL>
;Begin first loop that positions the cursor at the first record in block
;one (when ecount = 0).
$first
select 'x' from system.dual where
    :zero.ecount = 0
/
;Message if value not found :
$continue $loop
;Must value exist Y/N :
N
;Back cursor up one record and check for top record again.
$loop
#exemacro exetrg one_prvrec;
/
;Message if value not found :
$first $first
;Must value exist Y/N :
N
;Until the last record (when :one.carrier_code is null)
$continue
select 'x' from system.dual where
    :one.carrier_code is null
/
;Message if value not found :
$out $cont1
;Must value exist Y/N :
N
;Clear the record so that a commit will not try to commit to block one
;(the approved_carrier view) but will commit all records in block two.
$cont1
#exemacro clrrec;
/
;Message if value not found :
$continue $continue
;Must value exist Y/N :
N
;Commits all records held in block two, a hidden block, and moves
;the cursor to the select_or_enter field.
$out
#exemacro commit; exit;

```

```

AND
  (code_of_service_approved in(select code_of_service from conditions
where user_id = user and terminal_id = userenv('TERMINAL')
and analysis_seq_num =:header.analysis_set_num))
OR ( 0=(select count(code_of_service) from conditions where
  user_id = user
  and terminal_id = userenv('TERMINAL') and
  analysis_seq_num = :header.analysis_set_num) AND
  (:zero.category in ('ALL','OTHER','VM') or
  (:zero.category = 'INTL' and code_of_service_approved in
    ('4','5','6','7','8','J','T')) or
  (:zero.category = 'MH' and code_of_service_approved = 'S') or
  (:zero.category = 'INTRA' and
    code_of_service_approved in ('1B','2B'))
  or (:zero.category = 'INTER' and
    code_of_service_approved in ('1A','2A')) or
  (:zero.category = 'DOM' and
    code_of_service_approved in ('1A','1B','2A','2B')) or
  (:zero.category = 'DPM' and
    code_of_service_approved like 'B_' or code_of_service_approved
    like 'H_'))))
/
;Message if value not found :
$continue $stop Carrier code does not exist or does not match category.
;Must value exist Y/N :
N
;If the SCAC is not valid, display error message that the user must
;acknowledge. Clear, and let user begin again.
$stop
#exemacro pause; clrrec; gofld one.carrier_code;
/
;Message if value not found :
*$send $end
;Must value exist Y/N :
Y
;If the SCAC is appropriate, move to block two, execute bottom2 to
;position the cursor one past the last record held there.
$continue
#exemacro goblk two; exetrg bottom2;
/
;Message if value not found :

;Must value exist Y/N :
N
;Create this record in block two.
select user, userenv('TERMINAL'),1,:one.carrier_code,
'*,:one.carrier_name

```

```

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
*eentqry
;SQL>
;Check for entry of an illegal (blank) record.
select 'x' from system.dual where
:one.carrier_code is not null
/
;Message if value not found :
Blank records cannot be selected.
;Must value exist Y/N :
Y
;Check for an * (does user wish to insert or cancel this record).
select 'x' from system.dual where
:one.star = '*'
/
;Message if value not found :
$star1 $no_star
;Must value exist Y/N :
N
$star1
#exemacro exetry del_old_scac;
/
;Message if value not found :
$star $star
;Must value exist Y/N :
N
;If an * exists, then the user wants to cancel this entry.
;Deletes the record entered in block two and returns to block one.
$star
#exemacro goblk two; exetry top2; exetry search_and_destroy;
      gofld one.carrier_code; clrrec; gofld one.carrier_code;
/
;Message if value not found :
$end $end
;Must value exist Y/N :
N
;Displays the carrier name for the corresponding SCAC and validates
;whether the SCAC entered is appropriate for the service category and
;COS(s) the user has selected.
$no_star
select carrier_name into :one.carrier_name from approved_carrier where
      :one.carrier_code = carrier_code

```

```

into :two.user_id, :two.terminal_id, :two.analysis_seq_num,
      :two.carrier_code, :one.star ,:two.carrier_name from system.dual
/
;Message if value not found :
$get $get
;Must value exist Y/N :
N
$get
#exemacro goblk one; exetrg get_old_scac;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Postion cursor in block one.
$end
#exemacro gofld one.carrier_code;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
*get_old_scac
;SQL>
;This trigger selects historic SCAC(s) corresponding to the carriers the
;user selects and puts them in block two. From there they are entered
;in the conditions table.
$step1
#exemacro goblk three; exeqrg;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Check to see if there is an historic SCAC corresponding to the carrier
;the user has chosen.
$step2
select 'x' from system.dual where :three.old_carrier_code is null
/
;Message if value not found :
$end $step3
;Must value exist Y/N :
N
;Go to the first empty row in block two.
$step3
#exemacro goblk two; exetrg bottom2;

```

```

/
;Message if value not found :

;Must value exist Y/N :
Y
;Insert a record into block two.
$step4
select user, userenv('TERMINAL'), :header.analysis_set_num,
       :three.old_carrier_code into
       :two.user_id,      :two.terminal_id,      :two.analysis_seq_num,
       :two.carrier_code
       from system.dual
/
;Message if value not found :

;Must value exist Y/N :
Y
$step5
#exemacro goblk three;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Go to the record in block three that matches the record just entered
;in block two.
$step6
select 'x' from dual where :two.carrier_code = :three.old_carrier_code
/
;Message if value not found :

;Must value exist Y/N :
Y
$step7
#exemacro nxtrec;
/
;Message if value not found :
$step2 $step2
;Must value exist Y/N :
N
$end
#exemacro gofld :one.carrier_code;

;Message if value not found :

;Must value exist Y/N :
Y

```

;Increments the number in ecount and performs the NXTREC function.

;Field name :

*one_nxtrec

;SQL>

select :zero.ecount + 1 into :zero.ecount from system.dual

where :one.carrier_code is not null

/

;Message if value not found :

;Must value exist Y/N :

N

#exemacro nxtrec;

;Message if value not found :

;Must value exist Y/N :

Y

;Decrements the ecount counter and performs the PRVREC function.

;Field name :

*one_prvrec

;SQL>

select :zero.ecount - 1 into :zero.ecount from system.dual

where :zero.ecount != 0

/

;Message if value not found :

;Must value exist Y/N :

N

#exemacro prvrec;

;Message if value not found :

;Must value exist Y/N :

Y

;This trigger applies ONLY if the user has chosen to select SCAC(s) from
;a list.

;It commits records held in block two and moves the cursor to the

;select_or_enter field.

;Field name :

*scommit

;SQL>

#exemacro commit; exit;

;Message if value not found :

;Must value exist Y/N :

Y

```

;This trigger is called ONLY if the user is selecting SCAC(s) from a
;list.
;Field name :
*sentqry
;SQL>
;Checks for an illegal entry.
select 'x' from system.dual where
:one.carrier_code is not null
/
;Message if value not found :
Blank records cannot be selected.
;Must value exist Y/N :
Y
;Checks to see whether user is making a selection or cancelling a
;previous selection.
select 'x' from system.dual where
:one.star = '*'
/
;Message if value not found :
$star1 $no_star
;Must value exist Y/N :
N
$star1
#exemacro exetrg del_old_scac;
/
;Message if value not found :
$star2 $star2
;Must value exist Y/N :
N
;If the user is cancelling a selection, go to block two and delete that
;record.
$star2
#exemacro goblk two;
/
;Message if value not found :

;Must value exist Y/N :
Y
$star2b
#exemacro exetrg top2;
/
;Message if value not found :

;Must value exist Y/N :
Y
$star2c
#exemacro exetrg search_and_destroy;

```

```

/
;Message if value not found :

;Must value exist Y/N :
Y
;And delete the *.
select ' ' into :one.star from system.dual
/
;Message if value not found :
$end $end Step 3 of sentgry fails
;Must value exist Y/N :
Y
;If the user is selecting a SCAC, go to block two and position the cursor
;one record past the last record entered.
$no_star
#exemacro goblk two; exetrg bottom2;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Insert this new record.
select user, userenv('TERMINAL'), 1, :one.carrier_code, '*'
into :two.user_id, :two.terminal_id, :two.analysis_seq_num,
:two.carrier_code, :one.star from system.dual
/
;Message if value not found :
$get $get
;Must value exist Y/N :
N
$get
#exemacro goblk one; exetrg get_old_scac;
/
;Message if value not found :

;Must value exist Y/N :
N
;Go back to block one.
$end
#exemacro goblk one;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
carrier_name

```

```

;Type of field :
CHAR
;Length of field / Display length / Query length :
48 / 48 / 48
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
25
;Prompt :

;Allow field to be entered Y/N :
**N
;Allow entry of query condition Y/N :
N
;Hide value of field Y/N :
N
;SQL>

;Field name :
star
;Type of field :
CHAR
;Length of field / Display length / Query length :
1 / 1 / 1
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
19
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :

```

```
Y
;Allow entry of query condition Y/N :
N
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
Y
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
Use arrow keys to scroll up or down. Press END beside your choice(s).
;Lowest value :

;Highest value :

;Field name :
carrier_code
;Type of field :
CHAR
;Length of field / Display length / Query length :
4 / 4 / 4
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
1
;Column :
20
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
N
;Allow entry of query condition Y/N :
N
```

```

;Hide value of field Y/N :
N
;SQL>
**key-nxtrec
/
;SQL>
#exemacro exetrg one_nxtrec; gofld one.carrier_code;
/
;Message if value not found :

;Must value exist Y/N :
Y
;SQL>
**key-prvrec
/
;SQL>
#exemacro exetrg one_prvrec; gofld one.carrier_code;

;Message if value not found :

;Must value exist Y/N :
Y
;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
Y
;Help message :
Enter a valid SCAC and press END. Arrow keys move cursor.
;Lowest value :

;Highest value :

;Field name :

;Block zero holds the enterable fields on this screen. Control will
;branch based on whether the user wants to enter or list SCACs. If the
;user enters, control will pass to block one.
;If the user lists, control will pass to the sorting and the substring
;fields before passing to block one.
;Block zero is a control block with no base table.
;Block name / Description :
zero/zero
;Table name :

```

```

*
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1 / 5
;The check_carrier_code trigger, called from the sort field, in turn
;branches to execute different triggers based on the value contained in
;sortby.
;Field name :
*check_carrier_code
;SQL>
#exemacro case :zero.sort is
  when 'D' then exetrg put_carrier_name;
  when 'C' then exetrg put_scac;
  when others then exetrg s_error;
end case;

;Message if value not found :

;Must value exist Y/N :
Y
;This trigger is called ONLY if the user chooses to enter SCACs.
;Field name :
*enter_list
;SQL>
;Initialize the ecount, which keeps track of the position of the cursor.
#copy '0' :zero.ecount
/
;Message if value not found :

;Must value exist Y/N :
N
;Find all previous records for this session and user that exist in the
;conditions table.
#exemacro goblk two; exeqry;
/
;Message if value not found :

;Must value exist Y/N :
N
;Begin loop until record is not null.
$loop_top
select 'x' from system.dual where
  :two.user_id is not null
/
;Message if value not found :
$continue $stop

```

```

;Must value exist Y/N :
N
;Until record is null, insert/copy from block two, a hidden block, to
;block one.
$continue
select '*', :two.carrier_code, carrier_name into
:one.star, :one.carrier_code, :one.carrier_name
from approved_carrier_view
where :two.carrier_code = carrier_code
/
;Message if value not found :
$next_match $next_no_match
;Must value exist Y/N :
Y
;Continue by going to block one, advance the cursor to the next record,
;go to block two, advance the cursor to the next record then go back
;to top of loop.
$next_match
#exemacro goblk one; exetrg one_nxtrec; goblk two; exetrg two_nxtrec;
/
;Message if value not found :
$loop_top $loop_top
;Must value exist Y/N :
N
$next_no_match
#exemacro goblk two; exetrg two_nxtrec;
/
;Message if value not found :
$loop_top $loop_top
;Must value exist Y/N :
N
;When finished go back to block one and display the help message.
$stop
#exemacro gofld one.carrier_code; help;

;Message if value not found :

;Must value exist Y/N :
Y
;Field name :
*find_others
;SQL>
;This trigger is called when a substring search has been entered by the
;user. It deletes all records that have been inserted previously by this
;user during this session.
#exemacro goblk two; clrblk; exetrg del_recs; goblk one; clrblk;
goblk zero; exetrg from_list;

```

```

;Message if value not found :

;Must value exist Y/N :
Y
;The from_list trigger is called ONLY if the user has chosen to list
;SCACs.
;Field name :
*from_list
;SQL>
;Find all previous records for this session and this user that exist in
;the condition table.
#exemacro goblk two; exeqry;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Position cursor at next record in block two.
$continue
#exemacro exetrg two_nxtrec;
/
;Message if value not found :
$loop_bottom $loop_bottom
;Must value exist Y/N :
N
;Find last record in block two.
$loop_bottom
select 'x' from system.dual where
:two.user_id is not null
/
;Message if value not found :
$continue $stop
;Must value exist Y/N :
N
;Continue ...
;           The cursor is positioned one past the last record in block
;two so that new records may be inserted without overwriting those
;previously inserted. Go to block one and execute a query.
$stop
#exemacro gofld one.star; exeqry; gofld one.star; help;

;Message if value not found :

;Must value exist Y/N :
Y
;Stores CARRIER_NAME as the choice for sorting.

```

```

;Field name :
*put_carrier_name
;SQL>
select 'CARRIER_NAME' into zero.sortby from dual

;Message if value not found :

;Must value exist Y/N :
Y
;Puts 'CARRIER CODE' as the choice for sorting.
;Field name :
*put_scac
;SQL>
select 'CARRIER_CODE' into zero.sortby from dual

;Message if value not found :

;Must value exist Y/N :
Y
;Displays error message.
;Field name :
*s_error
;SQL>
#exemacro null;

;Message if value not found :
*You must enter a C or a D in this field.
;Must value exist Y/N :
Y
;Displays error message.
;Field name :
*se_error
;SQL>
#exemacro null;

;Message if value not found :
*Error in entry of select_or_enter.
;Must value exist Y/N :
Y
;Field name :
sortby
;Type of field :
CHAR
;Length of field / Display length / Query length :
12 / 12 / 12
;Is this field in the base table Y/N :
N

```

```
;Default value :  
  
;Page :  
  
;SQL>  
  
;Field name :  
ecount  
;Type of field :  
INT  
;Length of field / Display length / Query length :  
2 / 2 / 2  
;Is this field in the base table Y/N :  
Y  
;Is this field part of the primary key Y/N :  
N  
;Default value :  
  
;Page :  
  
;SQL>  
  
;Field name :  
select_or_enter  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
1 / 1 / 1  
;Is this field in the base table Y/N :  
N  
;Default value :  
  
;Page :  
1  
;Line :  
4  
;Column :  
50  
;Prompt :  
  
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
N  
;Hide value of field Y/N :
```

```

N
;Enables the EXIT function.
;SQL>
**KEY-EXIT
/
;SQL>
#exemacro exit;
/
;Message if value not found :

;Must value exist Y/N :
Y
;Performs validation check and navigates cursor depending on the value
;entered.
;SQL>
**key-nxtfld
/
;SQL>
select 'x' from system.dual where
:zero.select_or_enter in ('L', 'E')
/
;Message if value not found :
Valid values are L and E.
;Must value exist Y/N :
Y
#exemacro case :zero.select_or_enter is
  when 'L' then gofld zero.sort; help;
  when 'E' then goblk one; clrblk; goblk zero; exetrg enter_list;
  when others then exetrg se_error;
end case;

;Message if value not found :

;Must value exist Y/N :
N
;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
Y
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
Y
;Help message :
*Enter either an 'L' to list all valid carrier codes or an 'E' to enter
them.
;Lowest value :

```

```

;Highest value :

;Field name :
sort
;Type of field :
CHAR
;Length of field / Display length / Query length :
1 / 1 / 1
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
5
;Column :
71
;Prompt :

;Allow field to be entered Y/N :
*Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;Calls trigger that performs validation check.
;Positions the cursor for next field.
;SQL>
**KEY-NXTFLD
/
;SQL>
#exemacro exetrg check_carrier_code;
/
;Message if value not found :

;Must value exist Y/N :
Y
#exemacro gofld zero.string; help;

;Message if value not found :

;Must value exist Y/N :
Y

```

```
;Is field mandatory Y/N :  
N  
;Is field fixed length Y/N :  
N  
;Auto jump to next field Y/N :  
N  
;Convert field to upper case Y/N :  
Y  
;Help message :  
Enter a 'C' to sort by carrier codes or a 'D' to sort by the description  
(name)  
;Lowest value :  
  
;Highest value :  
  
;Field name :  
string  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
2 / 2 / 2  
;Is this field in the base table Y/N :  
N  
;Default value :  
  
;Page :  
1  
;Line :  
6  
;Column :  
56  
;Prompt :  
  
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
Y  
;Hide value of field Y/N :  
N  
;Performs different functions and triggers based on whether or not the  
;string field has been entered.  
;SQL>  
**KEY-NXTFLD  
/  
;SQL>
```

```

select 'x' from dual where :zero.string is null
/
;Message if value not found :
$query $continue
;Must value exist Y/N :
N
$query
#exemacro goblk one; clrblk; goblk zero; exetrg from_list;
/
;Message if value not found :
$exit $exit
;Must value exist Y/N :
N
$continue
#exemacro exetrg find_others;
/
;Message if value not found :
$exit $exit
;Must value exist Y/N :
N
$exit
#exemacro null;

;Message if value not found :

;Must value exist Y/N :
Y
;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
Y
;Help message :
Enter up to two characters for searching or leave blank and press END.
;Lowest value :

;Highest value :

;Field name :
rec_count
;Type of field :
INT
;Length of field / Display length / Query length :
3 / 3 / 3

```

```
;Is this field in the base table Y/N :
N
;Default value :

;Page :
2
;Line :
16
;Column :
1
;Prompt :

;Allow field to be entered Y/N :
**N
;Allow entry of query condition Y/N :
N
;Hide value of field Y/N :
N
;SQL>

;Field name :
category
;Type of field :
CHAR
;Length of field / Display length / Query length :
5 / 5 / 5
;Is this field in the base table Y/N :
N
;Default value :

;Page :
2
;Line :
18
;Column :
1
;Prompt :

;Allow field to be entered Y/N :
**N
;Allow entry of query condition Y/N :
N
;Hide value of field Y/N :
N
;SQL>

;Field name :
```

;The header block is a display-only control block. Its purpose is to
;show identification data (application name, user name, etc.).
;The identification data is copied into global variables from main_par
;and copied into this block from the global area.

;Block name / Description :

header/header

;Table name :

*

;Check for uniqueness before inserting Y/N :

N

;Display/Buffer how many records :

1 / 1

;Field name :

user_name

;Type of field :

CHAR

;Length of field / Display length / Query length :

26 / 26 / 26

;Is this field in the base table Y/N :

N

;Default value :

;Page :

1

;Line :

2

;Column :

3

;Prompt :

;Allow field to be entered Y/N :

**Y

;Allow field to be updated Y/N :

Y

;Allow entry of query condition Y/N :

Y

;Hide value of field Y/N :

N

;SQL>

;Is field mandatory Y/N :

N

;Is field fixed length Y/N :

N

;Auto jump to next field Y/N :

N

```
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :

;Field name :
app_name
;Type of field :
CHAR
;Length of field / Display length / Query length :
26 / 26 / 26
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
2
;Column :
33
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :
```

```
;Highest value :

;Field name :
service_category
;Type of field :
CHAR
;Length of field / Display length / Query length :
6 / 6 / 6
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
2
;Column :
64
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :

;Field name :
analysis_set_num
```

```
;Type of field :
CHAR
;Length of field / Display length / Query length :
1 / 1 / 1
;Is this field in the base table Y/N :
N
;Default value :

;Page :
1
;Line :
2
;Column :
74
;Prompt :

;Allow field to be entered Y/N :
**Y
;Allow field to be updated Y/N :
Y
;Allow entry of query condition Y/N :
Y
;Hide value of field Y/N :
N
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :

;Field name :

;This block retrieves historic SCACs corresponding to carriers chosen
;by the user. It is used when inserting historic SCACs into the
;conditions table, or when deleting them.
;Block name / Description :
*three/three
```

```
;Enter default WHERE and ORDER BY clause :  
where carrier_code = :one.carrier_code
```

```
;Table name :  
CARRIER_CHANGE_HISTORY  
;Check for uniqueness before inserting Y/N :  
N  
;Display/Buffer how many records :  
1 / 1  
;Field name :  
dummy  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
1 / 1 / 1  
;Is this field in the base table Y/N :  
N  
;Default value :
```

```
;Page :  
1  
;Line :  
2  
;Column :  
59  
;Prompt :
```

```
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
Y  
;Hide value of field Y/N :  
N  
;SQL>
```

```
;Is field mandatory Y/N :  
N  
;Is field fixed length Y/N :  
N  
;Auto jump to next field Y/N :  
N  
;Convert field to upper case Y/N :  
N  
;Help message :
```

```
;Lowest value :  
  
;Highest value :  
  
;Field name :  
old_carrier_code  
;Type of field :  
CHAR  
;Length of field / Display length / Query length :  
51 / 4 / 51  
;Is this field in the base table Y/N :  
Y  
;Is this field part of the primary key Y/N :  
N  
;Default value :  
  
;Page :  
3  
;Line :  
5  
;Column :  
19  
;Prompt :  
  
;Allow field to be entered Y/N :  
**Y  
;Allow field to be updated Y/N :  
Y  
;Allow entry of query condition Y/N :  
Y  
;Hide value of field Y/N :  
N  
;SQL>  
  
;Is field mandatory Y/N :  
N  
;Is field fixed length Y/N :  
N  
;Auto jump to next field Y/N :  
N  
;Convert field to upper case Y/N :  
N  
;Help message :  
  
;Lowest value :  
  
;Highest value :
```


INTERNAL DISTRIBUTION

1. F. P. Baxter
2. H. Bjerke
3. W. H. Cabage
4. T. Chiang
5. P. F. Daugherty
6. R. G. Edwards
7. T. L. James
8. J. O. Kolb
9. R. D. Kraemer
10. M. Kuliasha
11. R. S. Loffman
- 12-17. J. P. Loftis
18. V. Ng
19. R. L. Noe
20. D. E. Riechle
21. R. B. Shelton
22. P. T. Singley
23. P. M. Spears
24. J. N. Stone
25. L. F. Truett
26. D. P. Vogt
27. T. G. Yow
28. ORNL Patent Office
29. Central Research Office
30. Document Reference Section
- 31-33. Laboratory Records
34. Laboratory Records - RC

EXTERNAL DISTRIBUTION

35. Dr. Don Alvic, Senior Associate Director, Energy, Environment, and Resource Center, 10521 Research Drive, Suite 100, Knoxville, TN 37932
36. Dr. Bruce G. Buchanan, Department of Computer Science, University of Pittsburgh, 206 Mineral Industries Building, Pittsburgh, PA 15260
37. J. J. Cuttica, Vice President of Research and Development, Gas Research Institute, 8600 W. Bryn Mawr Avenue, Chicago, IL 60631
38. Steven W. Jebo, Military Traffic Management Command Headquarters, PPM, 5611 Columbia Pike, Falls Church, VA 22041-5050
39. D. E. Morrison, 333 Oxford Road, East Lansing, MI 48823
40. Dr. Allan Hirsch, President, Dynamac Corporation, The Dynamac Building, 11140 Rockville Pike, Rockville, MD 20852
41. Dr. Martin Williams, Professor, Department of Economics, Northern Illinois University, DeKalb, IL 60115
42. Office of Assistant Manager for Energy Research and Development, DOE/ORO, P.O. Box 2001, Oak Ridge, TN 37831-8600
- 43-52. Office of Scientific and Technical Information, U.S. Department of Energy, P.O. Box 62, Oak Ridge, TN 37831