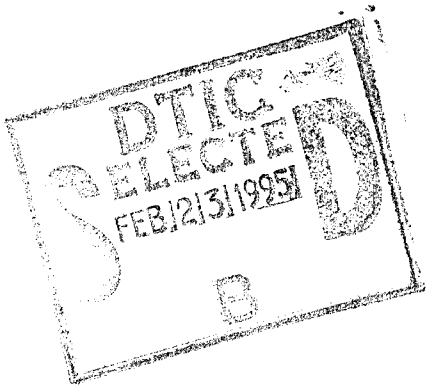
ARMY RESEARCH LABORATORY

Archival Information Management System

Timothy Paul Hanratty

ARL-TR-699 February 1995



NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute endorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1995	Final, June 1990 - June 1993

4. TITLE AND SUBTITLE

Archival Information Management System

5. FUNDING NUMBERS

4B592502350000

6. AUTHOR(S)

Timothy Paul Hanratty

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-CI-CA
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-OP-AP-L
Aberdeen Proving Ground, MD 21005-5066

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

ARL-TR-699

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

An invariable characteristic of any successful entity is its ability to effectively and efficiently manage its corporate information. The benefits gained as a result of maintaining such a resource have been echoed repeatedly through the years. Codifying the information and lessons learned from the past allow for the potential cost and time savings of the future.

Towards this end, the Ballistic Vulnerability Lethality Division (BVLD) of the U.S. Army Research Laboratory (ARL) has adopted a policy of strict configuration management for its computer models and associated data. One requirement of this new policy is that every finished analysis be marked by a complete audit trail of pertinent information. All requisite input and resultant data are to be archived to the extent that future reproducibility and interrogation of results will exist.

This report presents a prototype information management system named Archival Information Management System (AIMS), designed to meet the audit trail requirement for studies completed under the Modular Unix-Based Vulnerability Estimation Suite (MUVES) environment. Described is a system that combines the utility of a relational database management system with a traditional hierarchical file structure to produce a strategy to archive, recover, and interrogate information for all future analyses completed with MUVES.

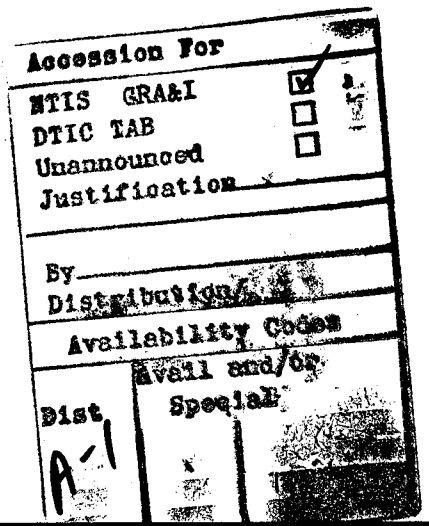
14. SUBJECT TERMS	15. NUMBER OF PAGES
configuration management, database, vulnerability analysis	51
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Intentionally Left Blank.

Acknowledgments

The author acknowledges with pleasure contributions by Fred Brundick and John Dumer. As original members of the Vulnerabiltiy Lethality Division Artificial Intelligence Research Team, their collaborative efforts on similar projects helped to resolve many of the theoretical and programming tasks.



Intentionally Left Blank.

Table of Contents

Intentionally Left Blank.

List of Figures

Intentionally Left Blank.

1. INTRODUCTION

The last 40 years have seen exponential growth in the type and amount of vulnerability/lethality (V/L)[a] information produced within the defense community. As a result of this growth, areas of expertise and valued sources of corporate information have evolved. To a large extent this evolution was driven by necessity and transpired under less than rigid configuration management and control. The majority of data produced and computer codes written during this period were traditionally managed on a very independent and individual basis. The general nature of this type of ad-hoc configuration management tended to result in unnecessary difficulties and inefficiencies. Data produced by one source was potentially unknown, inappropriately used, and/or inadvertently reproduced by a second source.

To combat the difficulties associated with ad-hoc configuration management, the Ballistic Vulnerability Lethality Division (BVLD) of the Survivability/Lethality Analysis Directorate (SLAD), U.S. Army Research Laboratory (ARL) made a conscientious effort to bring tighter configuration management and control to its methods of information analysis. Long held as the Army's authority for providing objective V/L assessment information for both foreign and domestic weapon systems, the "do more with less" policy of reduced funding and increased workload has made it increasingly incumbent upon the BVLD to adopt a stricter policy of insuring proper retainment and control over its valued corporate information.

Towards this end, a giant stride was made with the introduction of the Modular Unix-based Vulnerability Estimation Suite (MUVES) project. Designed as the vulnerability code of the future, MUVES provides an integrated software system that umbrellas not only the V/L methodologies of today under one suite, but more importantly allows for the controlled growth of tomorrow's methods as well.[1] The importance of the configuration management and control theme to the MUVES project cannot be over emphasized. In addition to providing an effective method of managing computer codes written in the future, the MUVES project provides the core around which future embellishments to all configuration management and control will evolve.

One embellishment that has been mandated and the subject of this report, is a system to automate an audit trail of information associated with each completed V/L analysis.[2] Required is a means to archive all requisite input and resultant data so that future interrogation and reproducibility of results can be attained. The requirement to better manage the data, in addition to the codes that produce the data, will provide the BVLD the necessary control of storing and disseminating information produced under the MUVES environment. Difficulties that could arise as to the availability and the applicability of results produced under this environment will be greatly reduced with this system.

---

[a]Defined in its simplest terms *vulnerability* is the quantitative assessment of a combat system's susceptibility to damage given a particular threat; while *lethality* is the measure of effectiveness with which an attacking weapon can inflict damage on a particular target.

Described in this report is the design and implementation of a prototype system, named "AIMS" - for Archival Information Management System, that couples the efficiencies of a database management system (DBMS) with the corporate information produced under the MUVES environment. The primary goal of the AIMS project is to complement the configuration management effort of the MUVES project by providing the required mechanism to catalog and retrieve all requisite input and resultant data produced for each completed study. Section 2 of this report provides an overview of the AIMS project, detailing the three subsystems that comprise the project. In section 3, the information flow that occurs from the time an analysis is completed through the archiving and recovery process is examined. Conclusions about the current state of the AIMS project and possible future embellishments are brought together in section 4.

2. SYSTEM OVERVIEW

At its highest level of abstraction AIMS can be viewed, as seen in Figure 1, as three distinct subsystems: an ***archive*** of vulnerability analysis information and two processes that manage this collection of data. The ***front-end process*** manages the information interrogation about the current state of the archive while the ***back-end process*** manages the information transfer between the archive and the MUVES environment.

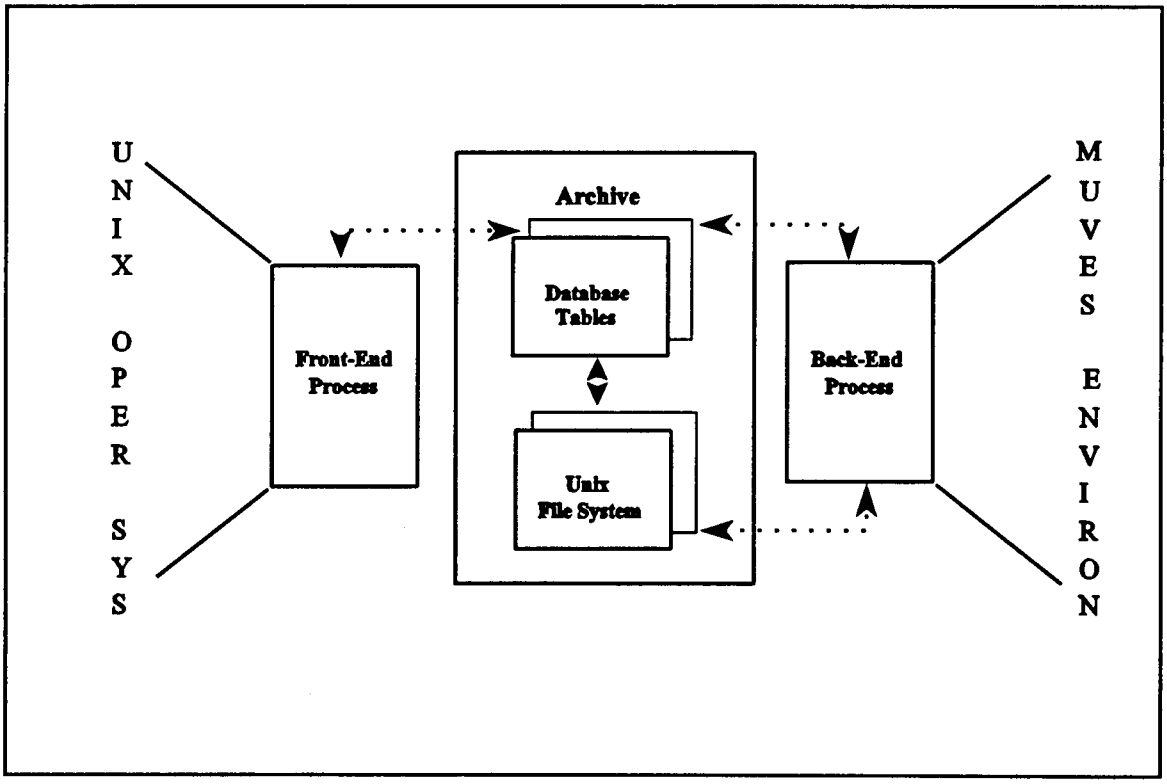


Figure 1: System Overview

2.1 The Archive

At the center of the AIMS system is the archive, an amalgamation of relational database tables with pointers to directories of MUVES's analysis information. By design every MUVES analysis is defined by a session file (see Appendix A: Sample Session File) which lists the requisite input files needed to derive the final results. Archiving the session file along with the corresponding input and resultant data files produced an ***official record*** of an analysis.[3]

AIMS organizes the archiving of an "official record" from a MUVES analysis into three connected components: a database component and two associate informational directory components, as shown in Figure 2.

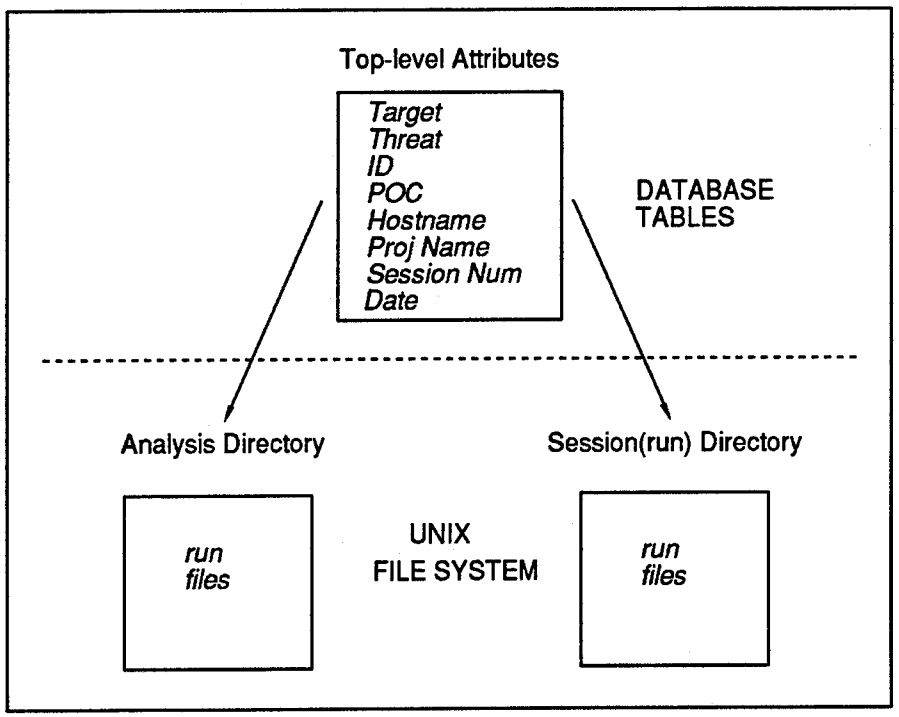


Figure 2: Archival Layout

The database component of the archive stores high-level characteristics of each completed analysis within the relational database management system of INGRES.[b] A database management system (DBMS) was chosen to manage this side of the archive because as a shared resource it offers an excellent means of managing large amounts of information in an

[b]All references to INGRES in this report correspond to Relational Technologies Inc., Version 6.2 INGRES.

organized and efficient manner. Data integrity, interrogation, and integration are greatly enhanced with a DMBS. The ***relational*** model database was selected, as opposed to a hierarchical or a network model, primarily for its availability and ease of use. Originally developed by Dr. E.F. Codd,[4] the relational model's stores data within the structure of a ***table***. Each table in turn is made up of ***rows*** and ***columns***, where a row can be thought of as a record of information (i.e., a completed analysis) and each column of the row defined by a particular attribute characteristic.

For AIMS, the column attributes, that define the top-level database table, were chosen for their inherent descriptive quality and are well suited for defining future queries about the archive. Each analysis that resides in the archive is uniquely defined by a system-generated primary attribute, the ***ID***entification number. The secondary attributes of ***target***, ***threat***, ***Point-of-Contact***, ***host name***, ***project name***, ***session number***, ***archive host,*** and ***date*** allow a varied ability of querying and may be modified in the future (Figure 3).

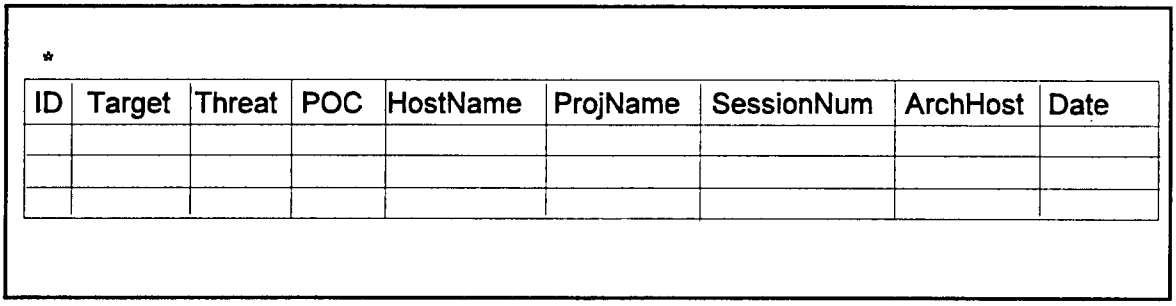


Figure 3: Column Attributes

The second and third components of the archive consist of two informational directories that utilize the existing hierarchical file structure. The Analysis Directory is used as the repository for all the information about a particular analysis that is needed to reproduce the results. This information is stored as a composite ***cpio***[c] file for each completed analysis. The Session Directory serves as an on-line augmentation to the top level database attributes and facilitates data interrogation queries by the front-end process. Files in both informational directories are linked to the database component via the system-generated primary ID. Functionality of the three connected components of the archive will be demonstrated later in the report.

2.2 The Front-End Process

Interrogation of the information contained within the archive, discussed in the previous section, is communicated to the end-user via the front-end process. The front-end process for AIMS is written utilizing the fourth generation language application generator provided by INGRES–Applications-By-Forms (ABF). While machine code, assembler language, and the

[c]cpio (copy archives in and out) is a convenient and portable method of archiving multiple directors of information.

high-level languages of FORTRAN, COBOL, and C represent the first three generations of computer languages, four generation languages are generally described as ***application generators***.[5] Application generators provide a level of improvement beyond the more conventional computer languages with an advanced ability for rapid prototyping and a robust set of operators that facilitate database access and screen input/output manipulation. Application developers regard the specialized qualities provided by the very high-level application languages as the programming tools of the future.

Defined in its simplest terms, an INGRES ABF application is composed of two basic components: the ***form*** and the ***menu operation list***. The form is the input/output medium supplying information to and conversely receiving information from a user. Information on the form is displayed as either static labels know as ***trim*** or dynamic attributes known as ***fields***. The menu operation list represents the group of different functions that can be performed within that form. It is the combination of a form with its associated menu operation list that defines the primary ABF building block known as the ABF ***frame*** shown in Figure 4.

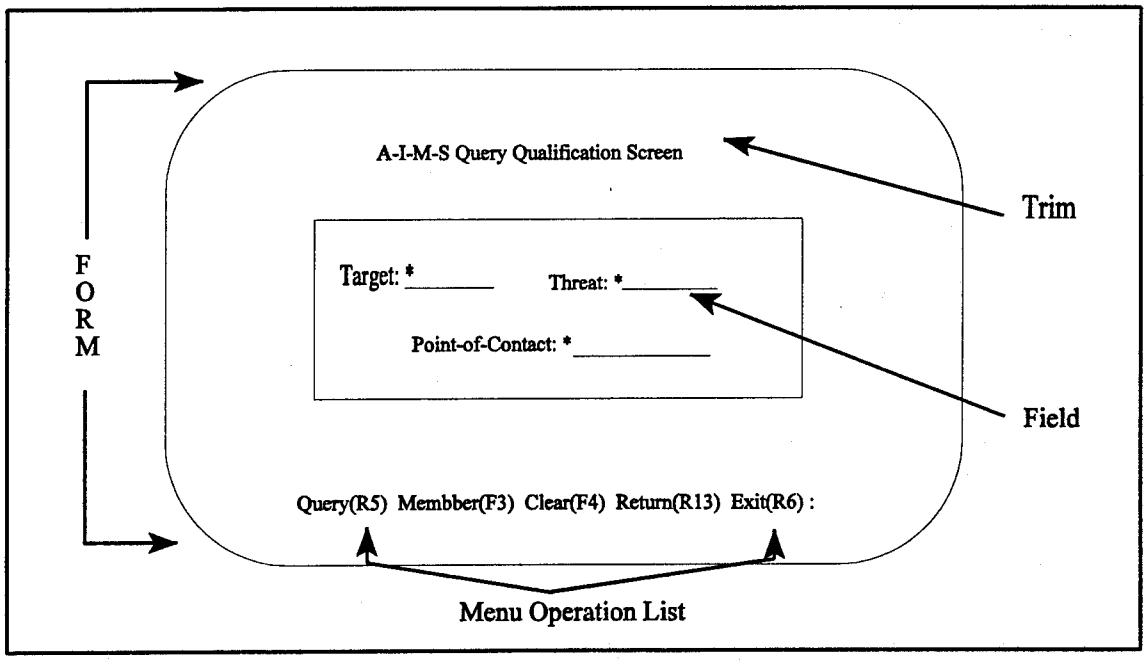


Figure 4: Application-By-Form Frame Composition

In general, each ABF frame has an individual task to accomplish. The hierarchical collection of ABF frames defines a particular ABF application. Shown in Figure 5 is the ABF frame structure definition for the AIMS front-end application. Examples of how management and analyst query the archive through the forms-based front-end process will be demonstrated in section 3.2. A complete listing of the AIMS ABF code is located in Appendix C: AIMS ABF Code - The Frontend Process.

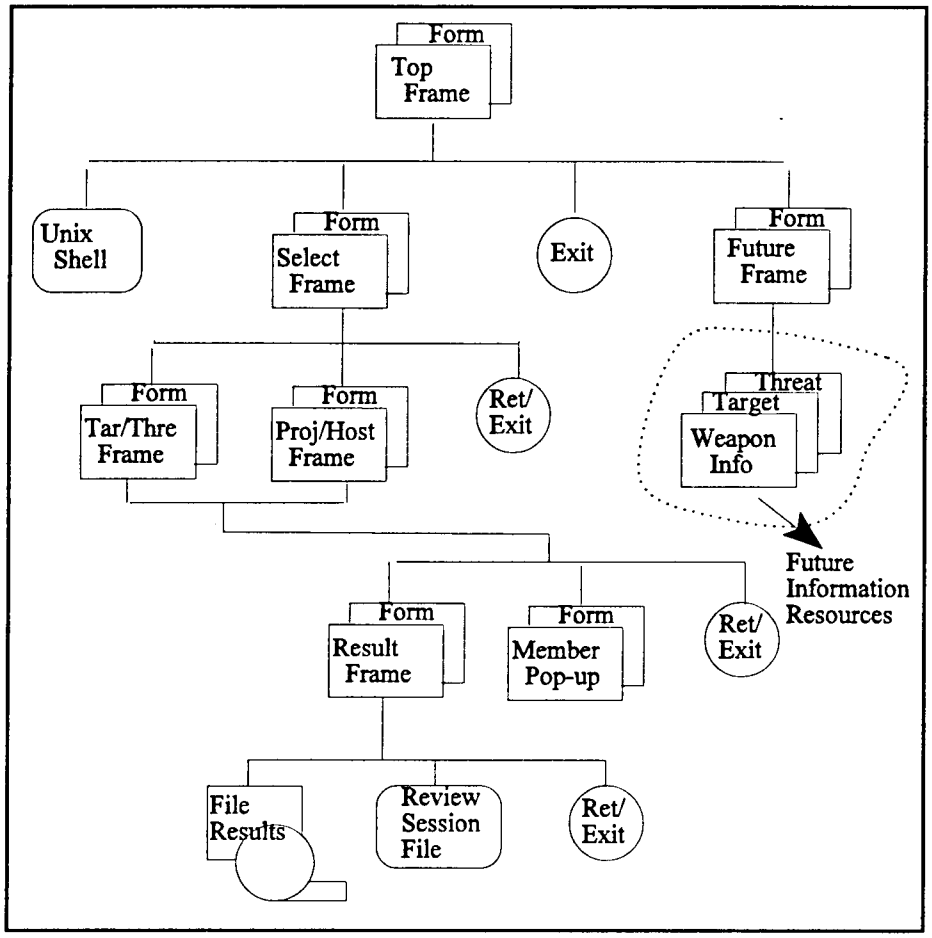


Figure 5: ABF Frame Structure

2.3 The Back-End Process

The responsibility of managing the transfer of information to the archive from the MUVES environment and, conversely, the transfer of information from the archive back to the MUVES environment rests with the back-end process. The exchange of information between the two entities must be reliable and exact. To accomplish this diverse task, the back-end process is coded as an embedded Structured Query Language (SQL) program within a host language of C and utilizes the MUVES Data Exchange (Dx) package. A complete listing of the back-end code is found in Appendix D: AIMS Embedded SQL Code - The Backend Process.

SQL was chosen over the other popular relational database query language, QUEL (Query Language), because in addition to supporting the standard database operations (including insert, delete, select and update), SQL has become the industry standard.[6] Embedding the database language of SQL within the procedural language of C combines the flexibility of a conventional language with the robust range of database management and manipulation language. This combined degree of pliancy was required to make the AIMS project work with cooperating subsystems.

The actual data communication between the MUVES environment and the back-end process is facilitated through the use of the MUVES Dx package. Developed for the MUVES project, the Dx package supports the bidirectional data exchange between cooperating master and slave processes. The back-end process of AIMS, which acts as the slave process, is initially activated by a special call to either ***archive*** or ***retrieve*** from the master MUVES user interface process. The special calls and the complete protocol for communications between the two processes can be found in Appendix B: MUVES DB (database) Protocol definition.

3. INFORMATION MANAGEMENT

The following section provides a general overview to how information is transferred between the major subsystems of AIMS. Discussed will be the information flow that occurs from the time an analysis is completed under the MUVES environment to the time the information is ***archived***, ***interrogated***, and ***recovered*** with the AIMS system. This overview is intended only to highlight the major features of the AIMS system.

3.1 Archiving

Upon the completion of a V/L assessment under the MUVES environment, the analyst is provided the opportunity to archive the project. By convention, the data associated with each MUVES analysis is contained within a unique directory under the MUVES file structure on its host machine (Figure 6). Once a determination has been made to archive this information, communication between the MUVES environment and the AIMS system is in order.

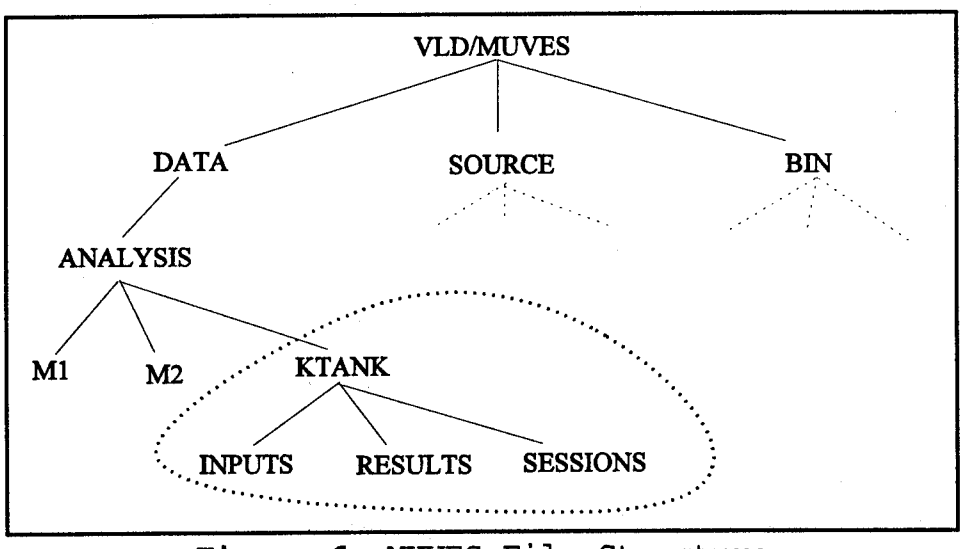


Figure 6: MUVES File Structure

From within the MUVES users interface, the analyst selects the administration and archiving menu options. Upon selection, MUVES initiates a connection with the AIMS back-end process

via the Dx protocol [see Appendix B: DB protocol]. To assure data synchronization and process control, the back-end process activates a file locking routine before any data is transferred . Tests are performed to assure the archive is accessible and the next available file position, system-generated ID, in the archive is ready.

Once ready, the back-end process receives data from MUVES in a predetermined order. First the top-level attributes are sent, followed by the session file, and, last, the composite cpio file of requisite input and resultant data. After receiving all of the information, the back-end process increments the primary counters to the archive and updates the database information appropriately. Successful completion by the back-end process is communicated back to the MUVES interface with the return transmission of the new unique archive ID number.

3.2 Interrogation

Once populated, the data in the archive is only as good as the method to examine it. AIMS answers this request with its front-end process. The front-end to AIMS provides the ability to effectively manage the interrogation of the current state of the AIMS archive with its forms-based interface. An example of this interface is demonstrated next with a series of screen dumps that would appear for a typical session with the AIMS front-end.

The opening form displayed by the AIMS front-end is shown in Figure 7. This form, along with its menu operation list, represents a typical menu selection frame for the AIMS front-end. Menu selection items are listed across the bottom of the form, while their descriptions are statically displayed above. The opening form has, in addition to the ***archival information*** menu item, entries for escaping to the shell, future expansion, and terminating the current session. Proceeding to the archival information frame is accomplished by selecting the "***archive***" menu option.

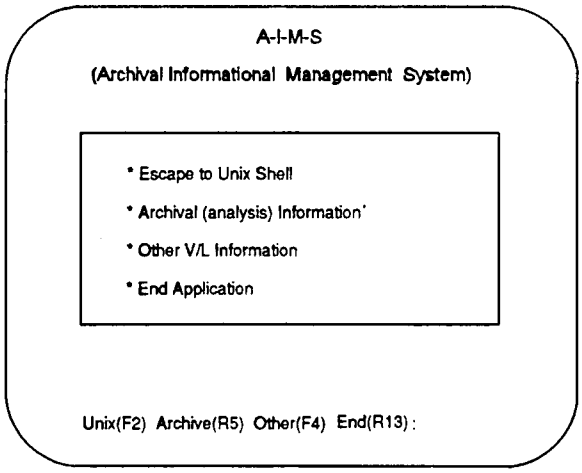


Figure 7: Opening AIMS Screen

The next form displayed in this example, shown in Figure 8, is of the archival query selection form. With this form a decision is made as to which of the two types of query selection modes to request: the ***target vs. threat*** query or the ***project vs. host*** query. Once selected, the appropriate query qualification screen will be displayed (Figure 9).

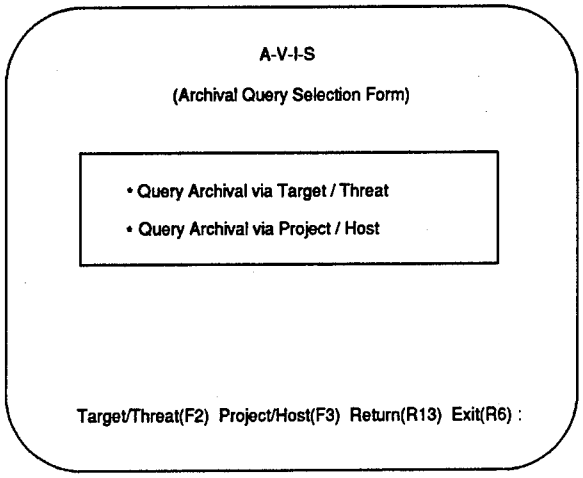


Figure 8: Query Selection Screen

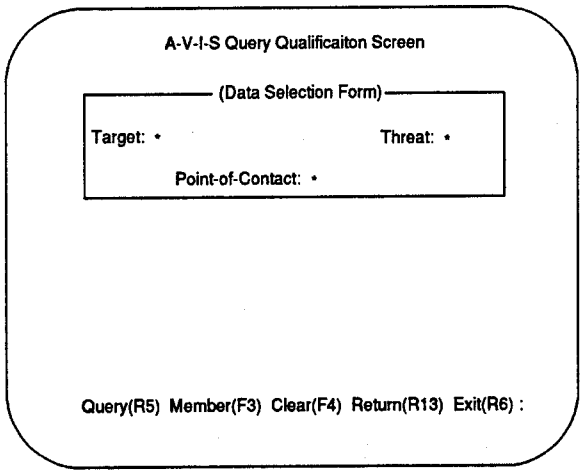


Figure 9:Query Qualification Screen

The query qualification form allows the user to enter the specifications about the pending query. Specifications to the attributes can be of either specific type (i.e., matching exactly), or of meta type, where the usual UNIX wild-card characters are allowed. In an effort to further assist the user in qualifying his query, the ***member menu*** operation is also available.

Demonstrated in Figure 10, the member operation displays the allowable range of entry for a particular attribute in question. Once satisfied with the query qualification, the user selects the ***query menu*** item to display the query display screen.

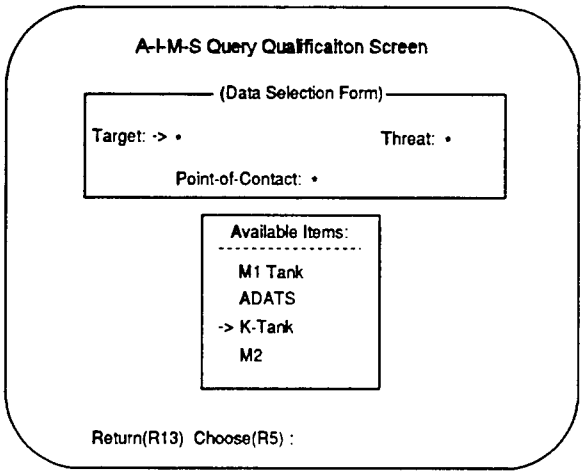


Figure 10: Member Selection Screen

The query display form shown in Figure 11 shows, in tabular form, the results of the given query. In our example, all information matching the specific target of a *K-tank* with *any* threat performed by *any* analyst is displayed. Further information about a particular combination can be found in that assessment's on-line session file, obtained through the ***sessioninfo menu*** operation. Hard copy of results can be directed to either a file or printing device.

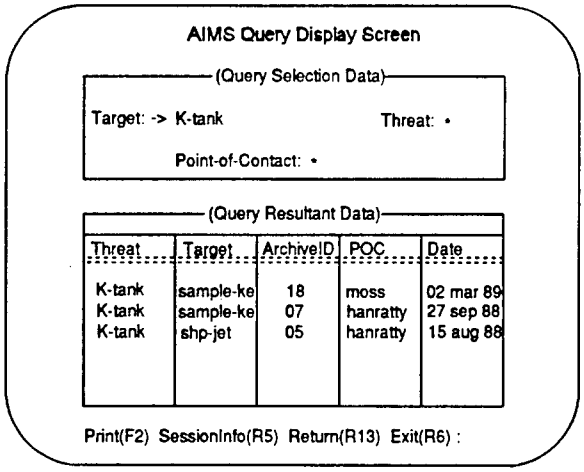


Figure 11: Query Result Screen

3.3 Recovery

Data is usually retrieved from the archive for one of three reasons: first, a project of similar characteristics has been requested and instead of starting from scratch, previous work is perused for commonalities; second, data that resided on a local host machine has been lost and needs to be recovered; or last, an analysis of past data has come under question and needs to be analyzed further. Regardless of the reason, once the recovery of data is requested, the recovery process must provide the requested data.

Similar to the archiving process, information recovery is performed as a connection of the MUVES environment with the AIMS back-end process. From within the MUVES user interface, the analyst selects the administration and restore menu options. The analyst is then prompted for the unique archive IDentification number in question. With an open Dx connection to the back-end process and a given archive IDentification number, the back-end process proceeds to query the database as to the validity of the request. If appropriate permission exists and the archive is available, the back-end process proceeds to execute and transmit data back to the requesting machine. Information is returned to the calling MUVES environment where a new directory is created to store the recovered information.

4. SUMMARY

Time and money spent researching or reinventing information from the past may not be available in the future. The requirement for effective and efficient information management is paramount. The AIMS prototype described in this report provides the necessary tool required to meet this challenge. With it, information produced under the MUVES environment can be safely archived for future interrogation and recovery. The predicted benefit in time and cost derived from the AIMS project managing the informational resource of future V/L assessments are vast.

Future enhancements to the AIMS prototype are limited only by ones imagination. Two immediate improvements should include the development of a direct database connection to the MUVES interface and improved mass storage capabilities. Where the interrogation and recovery process are now disjoint, the possibility exists to create a series of "canned" query operations that, if connected directly to the MUVES environment, would eliminate the two-step interrogation/recovery process. For mass storage improvements, the introduction of large optical disk units may prove to be necessary to house an ever growing archive in an efficient and cost effective manner.

Intentionally Left Blank.

5. REFERENCES

1. Philip J. Hanes, Scott L. Henry, Gary S. Moss, Karen R. Murray, Wendy A. Winner, "An Overview and Status Report of MUVES," BRL-MR-3679, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Md., July 1988.

2. David L. Rigotti, Paul H. Deitz, Donald F. Haskell, Michael W. Starks, Daniel P. Kirk, John R. Jacobson, Gilbert A. Bowers, Jill C. Smith, Gerald E. Mion, "Vulnerability/Lethality Division 5-Year Program Plan," BRL-SP-83, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Md., December 1989.

3. Philip J. Hanes, Scott L. Henry, Gary S. Moss, Karen R. Murray, Wendy A. Winner, "MUVES Vulnerabiliy Analyst's Guide," BRL-MR-3954, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Md., December 1991.

4. E.F. Codd, "A Relational Model of Data for Large Shared Data Banks." CACM 13, No. 6 (June 1970).

5. C.J. Date, An Introduction to Database Systems, Volume 1, Addison-Wesley Publishing Co., 1986.

6. Federal Information Processing Standards Publication, 1990 February 2, Number 127-1 (FIPS PUB 127-1).

Intentionally Left Blank

Appendix A:

Sample Session File

Intentionally Left Blank.

```
# sample session file
approx compart
target ml
eval inputs/des
threat p1165/initial/range0m
uplane 44 165 65 0000
aimvec 000-1000-11
aimvec 000-0.866025-0.500-11
aimvec 0000.5-0.86602500-11
aimvec 000-6.12323e-17-100-11
aimvec 0000.5-0.86602500-11
aimvec 0000.866025-0.500-11
aimvec 0001-1.22465e-1600-11
# below applies to evaluation
environ typical
mission firepower
environ typical
mission mobility
environ typical
mission catastrophic
environ threatened
mission internal_vol
environ threatened
mission gps
analyze 20
```

Intentionally Left Blank.

Appendix B:

MUVES DB Protocol

Intentionally Left Blank.

```
/**
DbProtocol.h–MUVES"Db"(database) protocol definition
        created:    91/01/3     GS Moss and TP Hanratty
        last edit:  %E%         GS Moss
        SCCS ID:    %W%

/**
        The Db package defines a protocol for communication between the MUVES user
        interface and a Dx slave process (server) whose function is to database MUVES runs
        and archive them to mass storage.
        DbHostName is a defined macro for the host name where the server executable
        lives and DbServerName is a defined macro for the path name of the server on
        DbHostName.
**/
#indef DbHostName
#define DbHostName              "vserv.brl.mil"
#endif
#ifndef DbServerName
#define DbServerName    "/vld/muves/bin/dbserver"
#endif
/**     Types of requests sent to the server:
        ____________________________________
        Packet identifier       Data type       Description
        ____________________________________
    Request to retrieve or archive initiates process **/

#define DbRetrieval     0x10        /*long(archive ID)*/
#define DbArchival      0x20        /*long(InfoRec)*/
```

```
        /*DbInfoRecord, DbRunFile, and DbTarFile follow*/
        DbInfoRecord
                DbTimestamp string from MUVES final results file
                DbHostName string host name
                DbProjectName string MUVES project name
                DbSessionNum long MUVES session no.
                DbRunNum long MUVES run no.
                DbTargetName string MUVES target name
                DbThreatName(*)string MUVES threat name
                DbPOC string user name
**/
#define DbInfoRecord      0x30
#define DbTimestamp       0x31  /*string(from final results file)*/
#define DbHostName        0x32  /*string(host name)*/
#define DbProjectName     0x33  /*string(MUVES project name)*/
#define DbSessionNum      0x34  /*long(MUVES session no.)*/
#define DbRunNum          0x35  /*long(MUVES run no.)*/
#define DbTargetName      0x36  /*string(MUVES target name)*/
#define DbThreatName      0x37  /*string(MUVES threat name)*/
#define DbPOC             0x38  /*string(user name)*/
/**
        DbRunFile file part of MUVES session file and DbTarFile follow
        (*)Multiple DbThreatName packets may be contiguous to accommodate runs with
        multiple threats.
**/

#define DbRunFile         0x40  /*file(part of MUVES session)*/
#define DbTarFile         0x50  /*file(package to archive)*/
```

```
/**     Types of responses sent from the server:
-------------------------------------------------------------------
Packet identifier          Data type      Description
-------------------------------------------------------------------
In response to successful receipt of archival request (server must spool files so that it can
guarantee their safety (modulo disk crashes) before responding):
                                                          **/
#define DbArchID           0x60           /*long(sequential archive ID)*/

/*In response to getting a retrieval request which matches an ID in the database:

#define DbAckRetrieve      0x70           /*long*/

/**     Once archive is retrieve from mass storage:
        DbInfoRecord
                DbTimestamp string YY/MM/DD HH:SS
                DbHostName string host name
                DbProjectName string MUVES project name
                DbSessionNum long MUVES session no.
                DbRunNum long MUVES run no.
                DbRunFile file part of MUVES session file
                DbTarFile file actual archive
**/
        In response to failed request:
        DbErrorCode long from $MUVES/include/sys/ErSym.h
**/
#define DbErrorCode 0x80 /*long(from $MUVES/include/sys/ErSym.h*/
```

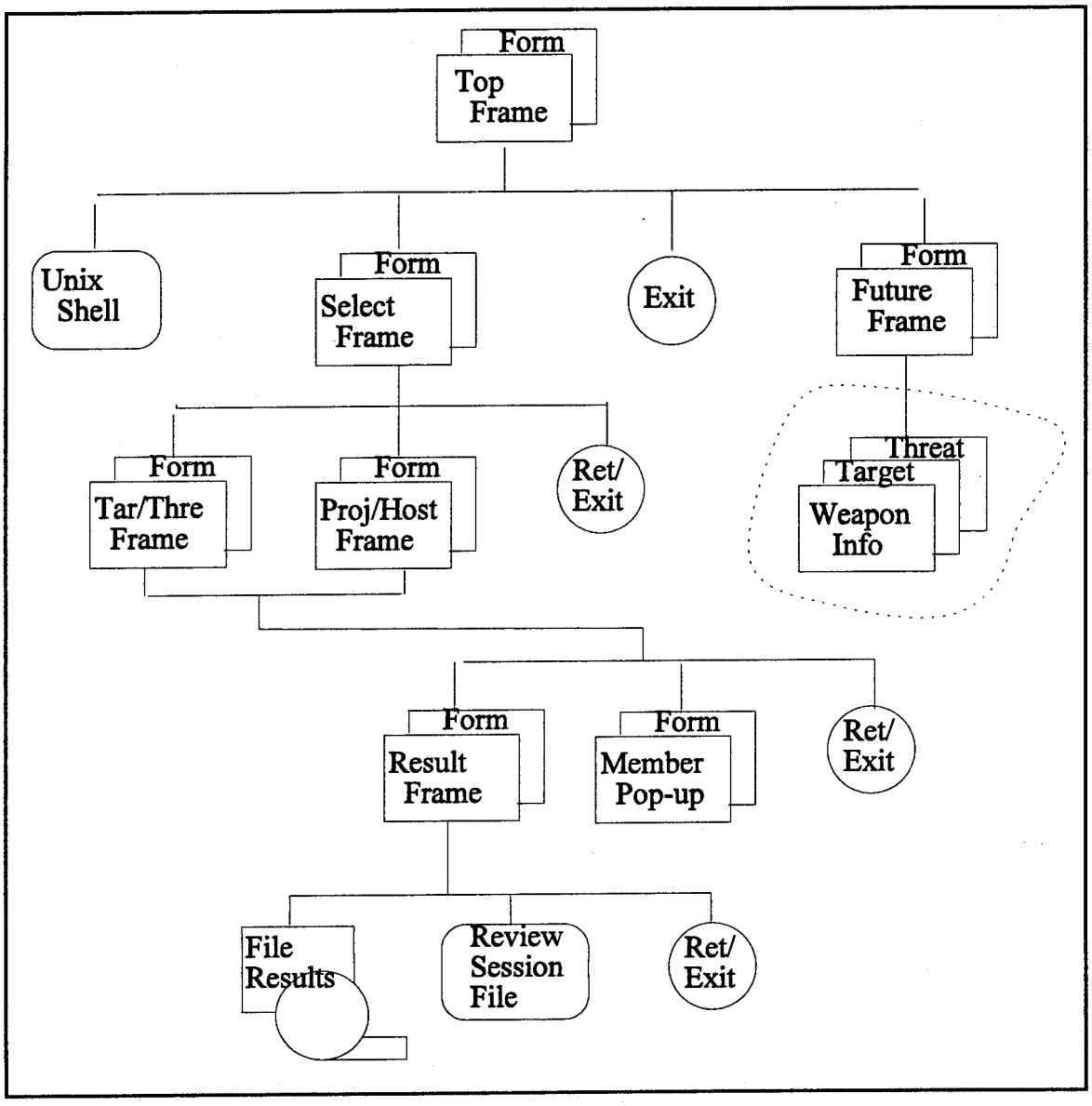
Intentionally Left Blank.

Appendix C

AIMS ABF Code Listing - The Front-end Process

Intentionally Left Blank.

AIMS Application-By-Form Structure Diagram



Intentionally Left Blank.

/**

Frame Name: Topframe

Frame Type: User-Defined

Description: Displays the top level Archival Informational Management System (AIMS) menu. This frame allows the user the ability to escape to the shell and affords a front-end for additional V/L information.

Form: topform

Relations: ?

Simple Fields: None

Table Fields: None

Initialization: Nominal

Field Activations: None

Key Activations:

Menu Options:

Unix	-	Escape to the Unix shell
Archive	-	Execute archiveframe for displaying archival info
Other	-	Hook for future V/L information hahaha
End	-	Exit this application

Called Frames: SelectFrame AIMS application
OtherFrame future V/L application

Special Issues: FSB function key layout

Revision History: 01/15/91 TPH created frame.

**/

```
/* Initial main menu items of AIMS */

'Unix  '=     begin
              call system;
              end

'Archive',    key frskey4 = begin
              callframe topqueryframe;
              end

'Other'=      begin
              callframe otherframe;
              end

'End',        key frskey3 = begin
              exit;
              end
```

```
/*****************************************************************
```

Frame Name:	OtherFrame
Frame Type:	User-Defined
Description:	This is a bullfeather frame that will allow future add ons to this extremely important application :)
Form:	Otherform
Relations:	called from TopFrame
Simple Fields:	none
Table Fields:	none
Initialization:	Nominal
Field Activations:	None
Key Activations:	

Menu Options:

Surrogate	-	hook to future surrogation application
GSB	-	hook to future GSB application
LTTB	-	hook to future LTTB application
LiveFire	-	hook to future LiveFire application
Return	-	return to calling frame

Called Procedures: none.

Called Frames: none ...

Special Issues: FSB function key layout ... and a number of popups

Revision History: 01/15/91 TPH created frame.

**********/

```
Surrogate = begin
        message 'The art of selecting the best from the worst ...' with style = popup;
        end

GSB   = begin
        message 'A future hook to other information ...' with style = popup;
        end

LTTB  = begin
        message 'A future hook to other information ...' with style = popup;
        end

LiveFire = begin
        message 'A future hook to other information ...' with style = popup;
        end

'Return', key frskey3 = begin
        return;
        end
```

/**********

Frame Name: SelectFrame

Frame Type: User-Defined

Description: This frame selects one of two query forms: Target vs. Threat or Project vs. Host.

Form: Selectform

Relations: called from TopFrame

Simple Fields: none

Table Fields: none

Initialization: nominal

Field Activations: none

Key Activations:

Menu Options:

Target/Treat	-	call the target/treat query frame
Project/Host	-	call the project/host query frame
Return	-	return to calling frame
Exit	-	exit program

Called Procedures: none.

Called Frames: none ...

Special Issues: FSB function key layout .

Revision History: 01/15/91 TPH created frame.

**********/

```
'Target/Threat' = begin
        callframe archiveframe;
        end;

'Project/Host' = begin
        callframe altqueryframe;
        end;

'Return', key frskey3 = begin
        return;
        end;

'Exit',  key frskey2 = begin
        exit;
        end
```

```
/*************************************************************************
```

Frame Name: TargetThreatFrame

Frame Type: User-Defined

Description: This frame is actually the beginning of the archival application. The user qualifies a query on the archive data base by narrowing the search to a target and/or theat combination.

Form: TargetThreatform

Relations: called from Selectframe

Simple Fields:

target	-	character 20
threat	-	character 20
point-of-contact	-	character 20

Table Fields: none

Initialization: Nominal

Field Activations: point-of-contact does a resume to target field

```
Key Activations:

Menu Options:

        Query       -       execute a query w/given data to Resultframe.
        Member      -       find available selection for current field/infoframe
        Clear       -       clear all field to the any/all character %
        Return      -       return to previous calling frame .. topframe
        Exit        -       exit application

Called Procedures: none.

Called Frames:      Resultframe
                    Memberframe

Special Issues:     FSB function key layout adopted.
                    should be noted that the '%' character is analagous
                    to the '*' under unix.

Revision History:   01/15/91      TPH    created frame.

**********************************************************************/

initialize(h_rows = smallint,CurField = char(80), NewVal = char(80),qual = char(80)) =
        begin
        redisplay;      h_rows = 0;
        message 'Enter appropriate information ... then select query'
                with style = popup (startrow=10);
        redisplay;      clear screen;
        resume field target;
        end

'Query', key frskey4= begin
        if target is null then target = '%'; endif;
        if threat is null then threat = '%'; endif;
        if poc    is null then poc    = '%'; endif;
        callframe archive2frame(archive2form.target       = :target;
                archive2form.threat  = :threat; archive2form.poc = :poc );
        resume field target;
        end
```

```
/*      if target is null then message 'aint no target'; sleep 5; endif;
        if threat is not null then set_forms field archiveform (underline (threat) = 1); endif;
        example of a subselection loop
        archiveform = select dateofanalysis = date,
                        target = target,
                        threat = threat,
                        archiveid = tarid,
                        sessionid = sessionid,
                        pointofcontact = poc from muvearchive
                where poc = :pointofcontact
        BEGIN OF SUBMENU
                begin
                initialize = begin
                        resume menu;
                        end
                'Next' = begin
                        next;
                        resume menu;
                        end
                'End'  = begin
                        endloop;
                        end
                end;
        clear field all;
        resume field target;
        end
*/

'Member'      = begin
/* this is where I am going to call a popup frame to display the current
fields available selection ... and passing back the last one sitting on */
        inquire_forms field archiveform (CurField = name);
        if CurField = 'target' then qual = 'target like ''' + :target + '''';
        elseif CurField = 'threat' then qual = 'threat like ''' + :threat + '''';
        else qual = 'poc like ''' + :poc + '''';
        endif;
        NewVal = callframe infoframe(infoform.CurField = :CurField;info.qual = qual)
with    style = popup;
        if CurField = 'target' then target = NewVal;
        elseif CurField = 'threat' then threat = NewVal;
        else poc = NewVal;
        endif;
        end
```

```
'Clear' =       begin
                clear field all;
                target = '%'; threat = '%'; poc    = '%';
                resume field target;
                end

'Return',       key frskey3 = begin
                return;
                end

'Exit',         key frskey2 = begin
                exit;
                end

field   'poc'   = begin
                resume next;
                end
```

/**********

Frame Name: ProjectHostFrame

Frame Type: User-Defined

Description: This frame is actually the beginning of the archival application. The user qualifies a query on the archive data base by narrowing the search to a project and/or host combination.

Form: ProjectHostform

Relations: called from Selectframe

Simple Fields:

project	-	character 20
host	-	character 20
sessionid	-	character 20

Table Fields: none

Initialization: Nominal

```
Field Activations:   point-of-contact does a resume to target field

Key Activations:

Menu Options:

        Query       -       execute a query w/given data to Resultframe.
        Member      -       find available selection for current field/infoframe
        Clear       -       clear all field to the any/all character %
        Return      -       return to previous calling frame .. topframe
        Exit        -       exit application

Called Procedures: none.

Called Frames:      Resultframe
                    Memberframe

Special Issues:     FSB function key layout adopted.
                    should be noted that the '%' character is analagous
                    to the '*' under unix.

Revision History:   01/15/91      TPH    created frame.

**********************************************************************/


initialize(h_rows = smallint,CurField = char(80),
          NewVal = char(80),qual = char(80)) = begin
        redisplay;
        h_rows = 0;
        message 'Enter appropriate information ... then select query'
                with style = popup (startrow=10);
        redisplay;
        clear screen;
        resume field project;
        end

'Query', key frskey4= begin
        if project is null then project = '%'; endif;
        if host is null then host = '%'; endif;
        if sessionid   is null then sessionid    = '%'; endif;
        callframe altquery2frame(altquery2form.project   = :project;
                                 altquery2form.host    = :host;
```

```
                                altquery2form.sessionid      = :sessionid );
        resume field project;
        end


'Member'      = begin

/* this is where I am going to call a popup frame to display the current
fields available selection ... and passing back the last one sitting on */

        inquire_forms field altqueryform (CurField = name);
        if CurField = 'project' then qual = 'projname like ''' + :project + '''';
        elseif CurField = 'host' then qual = 'hostname like ''' + :host + '''';
        else qual = 'sessionid like ''' + :sessionid + '''';
        endif;
        NewVal = callframe infoframe(infoform.CurField = :CurField;info.qual = qual)
                with style = popup;
        if CurField = 'project' then project = NewVal;
        elseif CurField = 'host' then host = NewVal;
        else sessionid = NewVal;
        endif;
                end

'Clear'         = begin
                clear field all;
                project = '%';
                host = '%';
                sessionid    = '%';
                resume field project;
                end

'Return',       key frskey3 = begin
                return;
                end

'Exit',         key frskey2 = begin
                exit;
                end

field   'sessionid'     = begin
                resume next;
                end
```

```
/*****************************************************************************

Frame Name:         ResultFrame

Frame Type:         User-Defined

Description:        The primary display of archival information dependent
                    on the user's query.

Form:               Resultform

Relations:          ?

Simple Fields:      target, threat, pointofcontact

Table Fields:       archive

Initialization:     a delicate select on a given query

Field Activations:  None

Key Activations:

Menu Options:

        Print       -       print the existing query info to a file
        Session     -       display the appropriate session file
        return      -       return to the calling frame
        End         -       Exit this application

Called Procedures:  Simple.sc .... used to print the query to a file

Special Issues:     FSB function key layout adopted.

Revision History:   01/15/91        TPH    created frame.

*****************************************************************************/

initialize(h_rows = smallint,
        tempfile=varchar(80),
        sessiondir=varchar(80)) = begin
/*      set_forms form (mode = 'read');             */
        sessiondir = '/sun/archive/runfile/';
```

```
        inittable archive read;
        archive2form.archive = select
                date = date,
                poc = poc,
                target = target,
                threat = threat,
                sessionid = sessionid
                from muvearchive
                where ( muvearchive.poc like :poc and
                        muvearchive.target  like :target and
                        muvearchive.threat  like :threat ) order by date desc;

        inquire_ingres(h_rows = rowcount);
        if h_rows = 0 then
        message 'No Match with given selection ...' with style = popup (startrow=10);
        return;
        endif;
        commit;
        end

'Print' = begin
 tempfile := PROMPT 'Enter File Name: ';
 if tempfile != '' then
        callproc simple(:tempfile,'target like ''' + :target + ''' and ' +
                        'threat like ''' + :threat + ''' and ' +
                        'poc   like ''' + :poc    + '''');
        clear screen;
        redisplay;
 endif;
end

'SessionInfo', key frskey4= begin
/* if needed the current location is known inquire_forms table ' ' (tempfile=column);  */
                helpfile 'SessionInfo' :sessiondir + :archive.sessionid;
                end

'return',       key frskey3= begin
                return;
                end

'exit',  key frskey2= begin
                exit;
                end
```

```
/*****************************************************************

Frame Name:         MemberFrame

Frame Type:         User-Defined

Description:        A slick way to allow the user the ability to peruse
                    available selection dependent on given string.

Form:               MemberForm

Relations:          ?

Simple Fields:      none

Table Fields:       list w/ one column name item.

Initialization:     a delicate select on given string.

Field Activations:  None

Key Activations:

Menu Options:

        Return      -       return to calling frame w/ character string '%'
        choose      -       return to calling frame w/ current item string

Called Frames:      none

Special Issues:     FSB function key layout

Revision History:   01/15/91      TPH    created frame.

*****************************************************************/

initialize(h_rows = smallint,CurField=varchar(80),qual=varchar(80)) = begin
/*              set_forms form (mode = 'read');             */
                inittable list read;
                if CurField = 'target' then
        list := select distinct target as items from muvearchive
                where :qual order by items;
                elseif CurField = 'threat' then
```

```
        list := select distinct threat as items from muvearchive
                where :qual order by items;
                elseif CurField = 'poc' then
        list := select distinct poc as items from muvearchive
                        where :qual order by items;
                elseif CurField = 'host' then
        list := select distinct hostname as items from muvearchive
                        where :qual order by items;
                elseif CurField = 'project' then
        list := select distinct projname as items from muvearchive
                        where :qual order by items;
                else
        list := select distinct sessionid as items from muvearchive
                        where :qual order by items;
                endif;
        inquire_ingres(h_rows = rowcount);
        if h_rows = 0 then
        message 'No Match with given selection ...' with style = popup (startrow=10);
        return '%';
        commit;
        endif;
end

'return',       key frskey3= begin
                return '%';
                end

'choose',       key frskey4= begin
                return list.items;
                end
```

Appendix D

AIMS Embedded SQL Code Listing - The Back-end Process

Intentionally Left Blank.

```
#include        <stdio.h>
#include        <unistd.h>
#include        <fcntl.h>
#include        <strings.h>

#include        </usr/muves/include/Dx.h>
#include        </usr/muves/include/Er.h>
#include        </usr/muves/include/DbProtocol.h>

#define MAXSTR          ((unsigned)256)
#define TIMEOUT         ((unsigned) 30)
#define InfoDir         "/usr/ing/ingres/MUVES/DB/"
#define CountFile       "/usr/ing/ingres/MUVES/DB/count"
#define DataBaseFile    "/usr/ing/ingres/MUVES/DB/DATABASE"
#define RunDirectory    "/usr/ing/ingres/MUVES/DB/RunDir/"

/* #define ArchiveDir     "/n/vim/sun/archive"
  #define TarDirectory   "/n/vim/sun/archive/tarfile"  */


#define DataBaseOn 1


FILE    *fp;
int     fd;


exec sql include sqlca;
exec sql begin declare section;
char    PacketId;
char    ArchiveDir[MAXSTR];
char    TarDirectory[MAXSTR];
char    ArchHostStr[MAXSTR];
char    TimeStr[MAXSTR];
char    tempstr[MAXSTR];
char    HostStr[MAXSTR];
char    ProjStr[MAXSTR];
char    TargetStr[MAXSTR];
char    *ThreatStr[MAXSTR];
long    ThreatCount=0;
long    SessionNum;
long    RunNum;
char    POCStr[MAXSTR];
char    filecountstr[MAXSTR];
long    pindex=0,Packet_Status=1,Data_Status=1,filecount;
long    NextPacketRecd[]={0x30,0x31,0x32,0x33,0x34,0x35,
                          0x36,0x37,0x38,0x39,0x40,0x50,0x60};
long    NextPacketSent[]={0x80,0x30,0x31,0x32,0x33,0x34,0x35,
                          0x40,0x50};
exec sql end declare section;
```

```
Err(mess)
char    *mess;
{
        (void)fputs("Dx_Slave: ",stderr);
        (void)fputs(mess,stderr);
        if (ErIsSet())
                {
                (void)fputs(":",stderr);
                ErPrint();
                ErClear();
                }
        else
                (void)putc('\n',stderr);
        Packet_Status=0;
        Data_Status=0;
}

main(argc, argv)
int     argc;
char    *argv[];
{

        register DxChannel      *chan;

/*      open initial connection to the master process */

        ErPrefix(ErSimple(argv[0]));

        if ((chan = DxOpen((char *)0, TIMEOUT)) == NULL) {
                Err("DxOpen failed");
                exit(1);
        }
        if (! DxInCharacter(chan,&PacketId)) {
                Err("DxInInteger failed");
                exit(1);
        }

/*      switch to appropriate routine dependint on packet recieved */

        switch( (int) PacketId) {

        case DbRetrieval:       ProcRetrieval(chan);
                                break;
        case DbArchival:        ErPLog("going to archive\n");
                                ProcArchival(chan);
                                break;
        default:            ErPLog("Illegal initial DbPacket Request");

        }
}
```

```
LockNFile()
{

/*
        used for process control ... allowing a single process to
        access the "filecount" file at a time.  LOCKF blocks (waits)
        til file is available.  filecountstr identifies the next
        position in the tar and session dir.  In addition check
        if NFS directory is available.
*/


        if (chdir(InfoDir) == -1) {
                (void) ErPLog("Error connecting to archive dir\nNOT Complete\n");
                exit(1);
        }
        if((fd = open(CountFile,O_RDWR)) == -1) {
                (void) ErPLog("Error opening filecount file\nNOT Complete\n");
                exit(1);
        }
        if (lockf(fd,1,0)) {
                (void) ErPLog("Error locking filecount file\nNOT Complete\n");
                exit(1);
        }
        if((fp=fdopen(fd,"r+")) == NULL) {
                (void) ErPLog("Error File Pointer to filecount file\nNOT Complete\n");
                exit(1);
        }
        fscanf(fp,"%s",filecountstr);
        filecount = atol(filecountstr);

}

UnLockFile(chan)
register DxChannel      *chan;
{

/*
        Unlock and update the filecount file to next position
        and report back to Master success (tarid)
*/

        ErPLog("unlocking files and sending back DbArchID\n");

        if (!DxOutCharacter(chan,DbArchID)) Err("DxOutInteger failed AckRec");
        else {
/*              had to force some hand-shaking here to make it work */
                if(!DxForceOut(chan))   Err("DxForceOutfailed");
                if ( !DxOutInteger(chan,filecount)) Err("DxOutInteger failed Unlockfile");
                if(!DxForceOut(chan))   Err("DxForceOutfailed");
        }
```

```
        filecount = filecount + 1;
        rewind(fp);
        (void) fprintf(fp,"%9d\n",filecount);
        lockf(fd,0,0);
        fclose(fp);
        close(fd);
        (void) ErPLog(" complete\n");
}

CleanUpMess(chan)
register DxChannel      *chan;
{
/*
        Erros have occurred ... so back out of everything
        and report back to Master failure
*/

        ErPLog("cleaning up mistake");
        if(!DxForceOut(chan))   Err("DxForceOutfailed");
        if (!DxOutCharacter(chan,DbErrorCode)) Err("DxOutInteger failed DbErrorCode");
        else {
/*              had to force some hand-shaking here to make it work */
                if(!DxForceOut(chan))   Err("DxForceOutfailed");
                if ( !DxOutInteger(chan,0)) Err("DxOutInteger failed Unlockfile");
                if(!DxForceOut(chan))   Err("DxForceOutfailed");
        }

        if ( !DxOutCharacter(chan,DbErrorCode)) Err("DxOutInteger failed ArchiveErr");
                if(!DxForceOut(chan))   Err("DxForceOutfailed");

        if (chdir(RunDirectory) == -1) {
                (void) ErPLog("possible error cleaning up\n");
        }
        else unlink(filecountstr);

        if (chdir(TarDirectory) == -1) {
                (void) ErPLog("possible error cleaning up\n");
        }
        else unlink(filecountstr);

        lockf(fd,0,0);
        fclose(fp);
        close(fd);
        ErPLog("NOT complete\n");
}


UpDateDB()
{
FILE    *tfp;
int             i;
```

```
char    timetmp[256];
char    monthtmp[256];

/*      needed to add II_SYSTEM to env ... lost with the rsh */

        putenv("II_SYSTEM=/usr/ing");
        exec sql connect test;
/*      exec sql grant all on muvearchive to public;*/

        ErPLog("ready to update database\n");
        ErPLog("Timestamp is %s\n",TimeStr);
        ErPLog("Hostname is  %s\n",HostStr);
        ErPLog("Projname is %s\n",ProjStr);
        ErPLog("Target   is %s\n",TargetStr);
        ErPLog("Threat   is %s\n",ThreatStr[0]);
        ErPLog("Session  is %d\n",SessionNum);
        ErPLog("Run      is %d\n",RunNum);
        ErPLog("Tarid    is %d\n",filecount);
        ErPLog("Sessid   is %s\n",filecountstr);
        ErPLog("ArchiveHost   is %s\n",ArchHostStr);

        strtok(TimeStr," ");
        strcpy(monthtmp,strtok(NULL," "));
        strcat(tempstr,strtok(NULL," "));
        strcat(tempstr,"-");
        strcat(tempstr,monthtmp);
        strcat(tempstr,"-");
        strcpy(timetmp,strtok(NULL," "));
        strcat(tempstr,strtok(NULL," ")+2);
        strcat(tempstr," ");
        strcat(tempstr,timetmp);
        ErPLog("the full string  is %s\n",tempstr);

        for (i=0;i<ThreatCount;i++) {
                exec sql insert into muvearchive
(date,poc,target,threat,sessionid,tarid,hostname,projname,sessionnum,runnum,archivehost)
values(:tempstr,:POCStr,:TargetStr,:ThreatStr[i],:filecountstr,:filecount,:HostStr,:ProjStr,:SessionNum,:Run
Num,:ArchHostStr);
        }

        if (sqlca.sqlcode < 0) {
                ErPLog("DataBase Update failed\n");
                exec sql rollback;
                exec sql disconnect;
                return(0);
        }

        exec sql disconnect;

        if ((tfp = fopen(DataBaseFile,"a+")) == NULL) {
                ErPLog("could not open flat DB file\n");
```

```
                return(0);
        }
        else {
                for (i=0;i<ThreatCount;i++) {
                fprintf(tfp,"%-3.3s %-10.10s %-10.10s %-3d %-3d
                        %-10.10s %-10.10s %-10.10s %-10.10s%-10.10s\n",
                        filecountstr, TargetStr, ThreatStr[i], SessionNum, RunNum,
                        POCStr, ProjStr, HostStr, TimeStr, ArchHostStr);
                }
                fclose(tfp);
                return(1);
        }

}

DBRet(filecount)
long    filecount;
{

/* first test to see if mass storage file system is on line */

ErPLog("Entered  embedded-DB retrieving procedure\n");


/* need to add II_SYSTEM to env .. lost with the rsh */

        putenv("II_SYSTEM=/usr/ing");

        exec sql connect test;

        exec sql select date,sessionid,hostname,projname,sessionnum,runnum
        into :TimeStr,:filecountstr,:HostStr,:ProjStr,:SessionNum,:RunNum
        from muvearchive where tarid = :filecount;

        if (sqlca.sqlcode < 0) {
                exec sql disconnect;
                (void) ErPLog("No Archive with that id\n");
                Data_Status = 0;
                return(0);
        }
        else {
                exec sql disconnect;

                strcpy(ArchiveDir,"/n/");
                strcat(ArchiveDir,ArchHostStr);
                strcpy(TarDirectory,ArchiveDir);
                strcat(TarDirectory,"/muves-arch/");

                if (chdir(ArchiveDir) == -1) {
                        (void) ErPLog("Error connecting to archive dir\n");
                        Data_Status = 0;
```

```
                return(0);
        }

        (void) ErPLog("Exiting DB Retrieval procedure\n");
        return(1);
    }
}



ProcRetrieval(chan)
register DxChannel      *chan;
{
    ErPLog("retrieving ...");
    if (!DxInInteger(chan,&filecount)) {
        ErPLog("DxInInteger failed on Retrieval Id Num\n");
        Packet_Status = 0;
    }
    else ErPLog("the filecount received is %d\n",filecount);

    if ( DBRet(filecount) ) {

        while (Packet_Status) {

        ErPLog("processing %d packetid\n",NextPacketSent[pindex]);
            switch(NextPacketSent[pindex]) {

                case DbAckRetrieve:   if ( !DxOutCharacter(chan,DbAckRetrieve))
                                          Err("DxOutCharacter failed DbAck");
                                      break;
                case DbInfoRecord:    if ( !DxOutCharacter(chan,DbInfoRecord))
                                          Err("DxOutCharacter failed DbInfo");
                                      break;
                case DbTimestamp:     if ( !DxOutCharacter(chan,DbTimestamp))
                                          Err("DxOutstart failed DbTime");
                                      if(!DxOutString(chan,TimeStr))
                                          Err("DxOutStringfail DbTime");
                                      break;
                case DbHostName:      if ( !DxOutCharacter(chan,DbHostName))
                                          Err("DxOutstart failed DbHost");
                                      if(!DxOutString(chan,HostStr))
                                          Err("DxOutStringfail DbHost");
                                      break;
                case DbProjectName:   if ( !DxOutCharacter(chan,DbProjectName))
                                          Err("DxOutstart failed DbProj");
                                      if(!DxOutString(chan,ProjStr))
                                          Err("DxOutStringfail DbProj");
                                      break;
                case DbSessionNum:    if ( !DxOutCharacter(chan,DbSessionNum))
                                          Err("DxOutCharacter failed DbSession");
```

```
                                        if ( !DxOutInteger(chan,SessionNum))
                                                Err("DxOutInteger fail DbSession");
                                        break;
                        case DbRunNum:          if ( !DxOutCharacter(chan,DbRunNum))
                                                Err("DxOutCharacter failed DbRun");
                                        if ( !DxOutInteger(chan,RunNum))
                                                Err("DxOutInteger fail DbRun");
                                        break;
                        case DbRunFile:         chdir(RunDirectory);
                                        if ( !DxOutCharacter(chan,DbRunFile))
                                                Err("DxOutCharacter failed DbRunfile");
                                        if(!DxOutFile(chan,filecountstr))
                                                Err("DxOutStringfail DbRunfile");
                                        break;
                        case DbTarFile:         chdir(TarDirectory);
                                        if ( !DxOutCharacter(chan,DbTarFile))
                                                Err("DxOutCharacter failed DbTarFile");
                                        if(!DxOutFile(chan,filecountstr))
                                                Err("DxOutStringfail DbTarFile");
                                /* needed to force the output ... slave was reading own buffer below */
                                        if(!DxForceOut(chan))   Err("DxForceOutfailed");
                                        Packet_Status = 0;
                                        break;
                        default:        Err("Sending Packet out of Range");
                        }
                pindex++;
                }
                if (Data_Status) ErPLog("complete\n");
                else ErPLog("Not complete\n");
        }
        else {
                if ( !DxOutCharacter(chan,DbErrorCode)) Err("DxOutChar failed");
                ErPLog("NOT complete\n");
        }

        if (!DxInCharacter(chan,&PacketId)) Err("DxInInteger failed to read DbClose");
        if (PacketId != DbCloseChannel) Err("Error in closing connection");
        if (!DxClose(chan)) Err("DxClose Failed");

}


ProcArchival(chan)
register DxChannel      *chan;
{
        int     i;
        ErPLog("archiving ... ");
        LockNFile();

        while (Packet_Status){
                ErPLog("processing %d packetid\n",NextPacketRecd[pindex]);
```

```
if (!DxInCharacter(chan,&PacketId)) Err("DxInChar failed");
if (PacketId != NextPacketRecd[pindex]) {
        if (PacketId != DbPOC) {
                Err("Packet out of sync");
                continue;
        }
        else  pindex++;
}

switch((int)PacketId) {
case DbInfoRecord:      break;
case DbTimestamp:       if (! DxInString(chan,TimeStr,MAXSTR))
                                Err("DxInString Timestamp failed");
                        break;
case DbHostName:        if (! DxInString(chan,HostStr,MAXSTR))
                                Err("DxInString HostName failed");
                        break;
case DbProjectName:     if (! DxInString(chan,ProjStr,MAXSTR))
                                Err("DxInString ProjectStr failed");
                        break;
case DbSessionNum:      if (! DxInInteger(chan,&SessionNum))
                                Err("DxInInteger SessionNum failed");
                        break;
case DbRunNum:                  if (! DxInInteger(chan,&RunNum))
                                Err("DxInInteger RunNum failed");
                        break;
case DbTargetName:      if (! DxInString(chan,TargetStr,MAXSTR))
                                Err("DxInString TargetStr failed");
                        break;
case DbThreatName:      ThreatStr[ThreatCount] = (char *)malloc(MAXSTR);
                        if (! DxInString(chan,ThreatStr[ThreatCount++],MAXSTR))
                                Err("DxInString ThreatStr failed");
                        --pindex;
                        break;
case DbPOC:             if (! DxInString(chan,POCStr,MAXSTR))
                                Err("DxInString POCStr failed");
                        break;
case DbArchHost:        if (! DxInString(chan,ArchHostStr,MAXSTR))
                                Err("DxInString ArchHostStr failed");
/* build archivedir and TarDir */
                        strcpy(ArchiveDir,"/n/");
                        strcat(ArchiveDir,ArchHostStr);
                        strcpy(TarDirectory,ArchiveDir);
                        strcat(TarDirectory,"/muves-arch/");
                        break;
case DbRunFile:                 chdir(RunDirectory);
                        if (! DxInFile(chan,filecountstr))
                                Err("DxInFile RunFile failed");
                        break;
case DbTarFile:                 if (chdir(ArchiveDir) == -1) {
                (void) ErPLog("Error connecting to archive dir\nNOT Complete\n");
```

```
                                        exit(1);
                                }
                                if (! DxInFile(chan,filecountstr))
                                        Err("DxInFile TarFile failed");
                                Packet_Status = 0;
                                break;
                default:        Err("Packet out of Range");
                }
                pindex++;  /* set to next packettype */
        }

/* if data received is ok then update DB and unlock files */


if (Data_Status && UpDateDB()) UnLockFile(chan);
else CleanUpMess(chan);

/* read the closing packet and shutdown shop */

if (!DxInCharacter(chan,&PacketId)) Err("DxInInteger failed");
if (PacketId != DbCloseChannel) Err("Error in closing connection");
if (!DxClose(chan)) Err("DxClose Failed");

}
```

NO. OF COPIES	ORGANIZATION
2	ADMINISTRATOR ATTN DTIC DDA DEFENSE TECHNICAL INFO CTR CAMERON STATION ALEXANDRIA VA 22304-6145
1	COMMANDER ATTN AMCAM US ARMY MATERIEL COMMAND 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	DIRECTOR ATTN AMSRL OP SD TA US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR ATTN AMSRL OP SD TL US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR ATTN AMSRL OP SD TP US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
2	COMMANDER ATTN SMCAR TDC US ARMY ARDEC PCTNY ARSNL NJ 07806-5000
1	DIRECTOR ATTN SMCAR CCB TL BENET LABORATORIES ARSENAL STREET WATERVLIET NY 12189-4050
1	DIR USA ADVANCED SYSTEMS ATTN AMSAT R NR MS 219 1 R&A OFC AMES RESEARCH CENTER MOFFETT FLD CA 94035-1000

NO. OF COPIES	ORGANIZATION
1	COMMANDER ATTN AMSMI RD CS R DOC US ARMY MISSILE COMMAND REDSTONE ARSNL AL 35898-5010
1	COMMANDER ATTN AMSTA JSK ARMOR ENG BR US ARMY TANK AUTOMOTIVE CMD WARREN MI 48397-5000
1	DIRECTOR ATTN ATRC WSR USA TRADOC ANALYSIS CMD WSMR NM 88002-5502
1	COMMANDANT ATTN ATSH CD SECURITY MGR US ARMY INFANTRY SCHOOL FT BENNING GA 31905-5660
	ABERDEEN PROVING GROUND
2	DIR USAMSAA ATTN AMXSY D AMXSY MP H COHEN
1	CDR USATECOM ATTN AMSTE TC
1	DIR USAERDEC ATTN SCBRD RT
1	CDR USACBDCOM ATTN AMSCB CII
1	DIR USARL ATTN AMSRL SL I
5	DIR USARL ATTN AMSRL OP AP L

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

14 DIR USARL
ATTN: AMSRL-CI, W MERMAGEN, SR
AMSRL-CI-C, J GANTT
AMSRL-CI-CA,
E BAUR
J DUMER
T HANRATTY
R HELFMAN
AMSRL-CI-S, M TAYLOR
AMSRL-CI-B, P DIETZ
AMSRL-SL-BA,
J WALBERT
L ROACH
AMSRL-SL-BG, S SHEROKE
AMSRL-SL-BL,
D BELY
M FERRY
AMSRL-SL-BV, W MERMAGEN

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-TR-699 Date of Report February 1995

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001, APG, MD

Postage will be paid by addressee

Director
U.S. Army Research Laboratory
ATTN: AMSRL-OP-AP-L
Aberdeen Proving Ground, MD 21005-5066