AUDITRE

A GENERALIZED AUDITING FACILITY

Note: All references to the Auditre version in this manual are indicated by *vrs* or *v.r.s.* The current release of Auditre is version 5.2.1.

Note:	Throughout this manual, references to AUDITRE imply AUDITRE7 and/or AUDITRE8.

Comments pertaining to this document and the AUDITRE product are encouraged. Please send any comments in writing to:

Treehouse Software, Inc.
2605 Nicholson Road, Suite 1230
Sewickley, PA 15143
Phone: 724.759.7070
Fax: 724.759.7067
Email: support@treehouse.com
http://www.treehouse.com

Worldwide marketing of AUDITRE and other Treehouse products is handled through the Sewickley office.

Last Updated: February 18, 2016

This page intentionally left blank.

This Reference Manual explains the functions and capabilities of AUDITRE Version v.r.s. This manual describes:

- ADABAS Protection Log Processing
- AUDITRE Parameters
- Installation and Operations

The appendices contain many samples of parameters and resulting reports.

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF FIGURES

This page intentionally left blank.

SECTION I

INTRODUCTION

I.1 Introduction To AUDITRE

AUDITRE, a product of Treehouse Software, is a powerful, generalized auditing system for users of the ADABAS database management system. This document describes the functions and capabilities of AUDITRE including:

- The benefits and capabilities of AUDITRE for the ADABAS DBA, NATURAL Administrator, Data Center Operations Staff, Applications Programmers, Systems Analysts, DP Management, and End Users
- How to produce Audit reports from the ADABAS Protection Log
- How to install and use AUDITRE

The Need for a Standardized ADABAS Auditing Facility

ADABAS files can be updated from a variety of sources, such as NATURAL programs, Direct Calls, ADABAS native SQL, etc. This makes ADABAS very flexible, but introduces a serious weakness. The weakness is that there is no standardized method for auditing changes to the database from these various sources. The introduction of a standardized auditing capability could provide many benefits to all levels of the organization, including auditors, system designers and developers, programmers, database administrators, end users, and data processing managers.

The ADABAS Auditing Challenge

For the organization's Auditors, the biggest challenge is probably related to compliance testing. Compliance testing requires the Auditor to learn if the proper update procedure is being used in an application to update the records in a file. This is difficult in ADABAS because there is no method for automatically generating standard audit data. Without a record of exactly what changes an application program is making to the database, it is very difficult to test for compliance to the proper update procedure. This challenge has led some sites to invest a significant amount of time and resources in the development of Embedded Audit Routines (EARs).

Embedded Audit Routines are not the Solution

Adding EARs to application software is one way to get the data that Auditors need to do their job effectively. The data created by EARs is stored in special audit files on the system and used later to create audit reports. However, EARs can cause more harm than good.

For data processing management, the introduction of these EARs increases processing overhead and DASD storage requirements, which hampers the efforts of the DP department to function at maximum efficiency. The EARs add to overhead by requiring applications to perform additional audit logic to generate and store the audit data each time an update is made. The update data that these routines generate consumes space on the system until it is archived or erased.

For Security personnel, the use of EARs implies that audit data generated by an application is only as reliable as the programmer who codes the audit routines. A dishonest programmer could force a program to generate "legitimate" audit data when "incorrect" or illegal transactions are taking place.

For the programmer who is assigned the task of producing the audit reports, EARs are especially annoying. Some files, such as the PAYROLL-MASTER file, may be updated by the Payroll Department when annual cost-of-living increases are instituted or by Personnel

when an employee receives a pay raise. The programmer whose job it is to produce the audit report for the PAYROLL-MASTER file will have to access the update files for both the Payroll and Personnel systems, each of which may have a format and content that is completely different. The report program will have to deal with these different data formats and make sense of them on an audit report. In some cases, such as a Real-time Inventory Control system for a retail seller, the programming task could be monumental (and very expensive). Clearly, there must be a better way. The ADABAS Protection Log (PLOG) provides this better way.

The Impact of the Protection Log on ADABAS Auditing

When Protection Logging is turned on, ADABAS automatically generates a log of all changes to the database as they are made. The intended function of the Protection Log is for Backout/Regeneration of updates (i.e., protection of the database). AUDITRE uses this information for auditing, since the PLOG provides several benefits.

The Protection Log eliminates some of the programming difficulties caused by different applications updating the same database files. Information on updates made by Payroll and Personnel to the same file will be stored in the same place and in the same format. Creating an audit report will be easier and faster than it might have been with EARs.

Furthermore, use of the Protection Log adds little overhead to the various application systems and no overhead if Protection Logging is currently in use. Since the Protection Log is stored in a compressed format and no redundant data is generated (as would be the case with EARs when Protection Logging is on), storage requirements for audit data are significantly reduced.

The security and reliability of audit data is also improved. Using the Protection Log as the source of audit data practically guarantees that the audit reports are generated from reliable, complete, and unmodified data. The audit data will not have been generated by the applications' logic nor be subject to a particular programmer's view of what is sufficient.

Because the Protection Log contains information about all updates to all files from all sources at all times, the use of the Protection Log by AUDITRE provides an added benefit over EARs. The Auditor can use the Protection Log data to audit any application at any time without the knowledge of those who wrote or use the application software. Thus the Auditor can perform spot checks on any critical or trouble areas of the database as needed.

Finally, since the Protection Log is archivable, audit data can be maintained off-line indefinitely and recalled as needed. If it is discovered that a fraud had been going on for months, all the necessary data to investigate back to the beginning of the fraud would be available on the archived Protection Logs.

AUDITRE Function

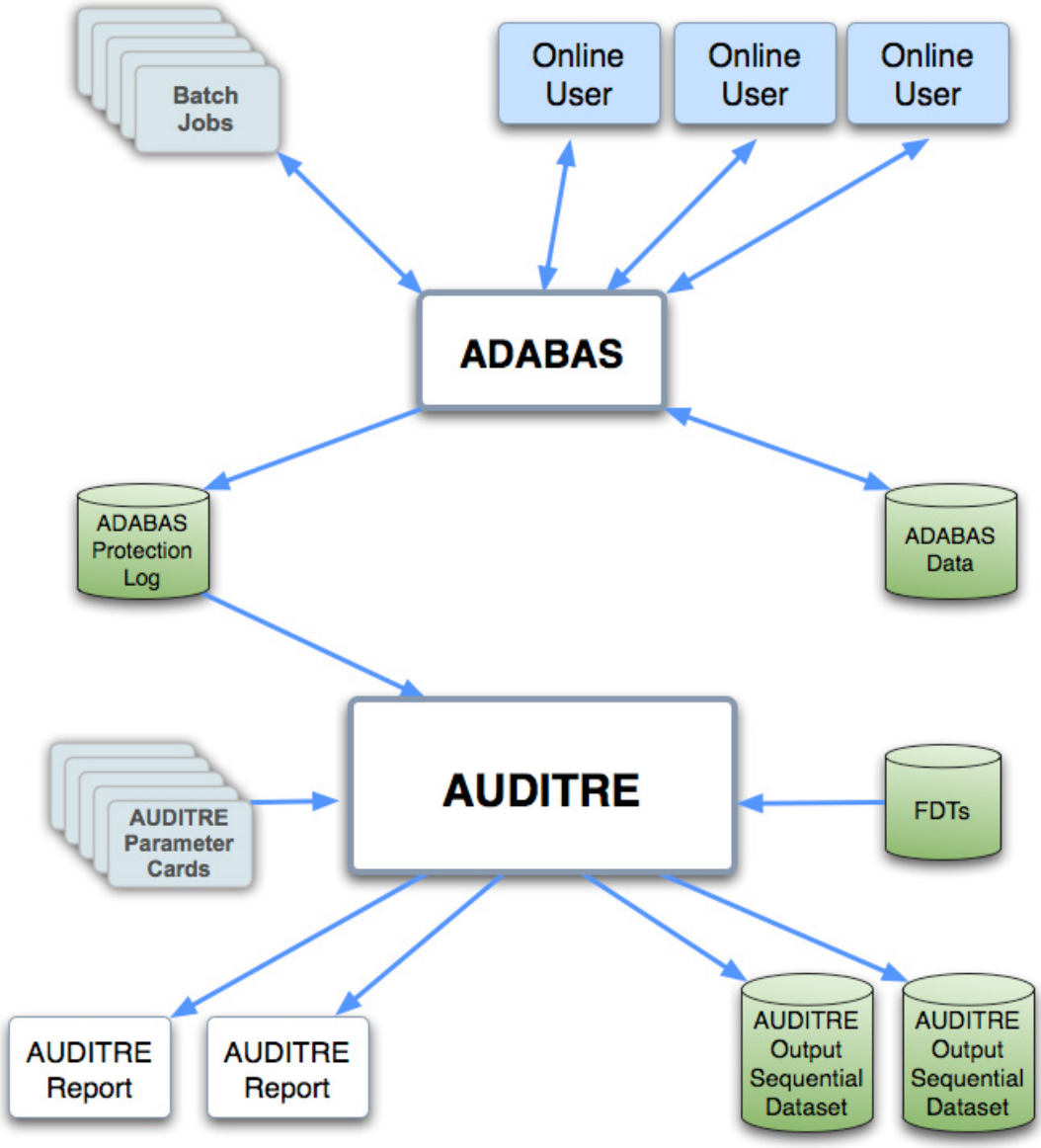
When an ADABAS transaction is made by a batch job **(A)** or an on-line user **(B)**, ADABAS **(C)** updates the physical database components **(D)** and writes information about the transaction to the ADABAS Protection Log **(E)**.

Since the information on the Protection Log **(E)** can be voluminous and is not in an easily readable form, AUDITRE **(F)** can be used to process the Protection Log data.

Taking direction from the user through AUDITRE parameter cards **(G)** and input from the Protection Log **(E)** and the Field Definition Table **(H)**, AUDITRE **(F)** is able to determine the number and kinds of transactions (Adds, Updates, or Deletes) existing on the ADABAS Protection Log(s) and produce multiple reports **(I)** or output the results to a sequential dataset(s) **(J)** for later processing in one pass.

Refer to Figure 1 on the next page.

FIGURE 1 - AUDITRE FUNCTIONAL PARTS



AUDITRE Provides the Best Auditing Solution

AUDITRE is completely self-contained and requires no additional support systems, such as extra programs, "audit exit routines", or specialized files. Audit reports can be generated on other CPUs at remote sites for security, if desired.

Furthermore, AUDITRE requires minimal installation and training time for the DBA and other users. Creation of new reports requires coding of only a few simple parameter statements. For instance, a report to identify all updates to certain fields on file 1427 would require coding a simple parameter statement, such as:

```
AUDIT     AA*,BB,DD,CC,FNR=1427
```

Changing the contents of a report requires only a few parameter changes, and adding new reports (or deleting old ones) is simple as well.

AUDITRE is designed in such a way that reports can be based on files, fields, sets of fields, and across logically related updated files. Reports contain only the files and fields desired. Multiple reports can be simultaneously generated with one pass of the Protection Log. This way the Auditor can use AUDITRE to produce clear, organized reports of database modification activity, including only the information desired.

For example, if a routine examination of the organization's cancelled checks shows that certain employees are being paid based on an incorrect HOURLY-RATE illegally altered by the employee, the Auditor might want to use the system to monitor the PAYROLL-MASTER file for future update activity on the HOURLY-RATE field.

AUDITRE is capable of monitoring changes to NATURAL programs. This can help by serving as a warning in the case of unauthorized changes and as a verification that authorized changes have been made. In this way, AUDITRE will further promote the integrity and safety of the ADABAS-related applications.

For maximum long-term benefit, selected data or reports can be archived to tape or disk for later processing. This allows audits to take place "after the fact" for certain critical systems if a problem is later discovered. It will also give Auditors confidence that applications still perform updates correctly after the applications have been upgraded.

AUDITRE Makes ADABAS Auditing Easier

To clearly see how AUDITRE can make routine audits easier, consider a situation where the Protection Log contains over one million records. These records are blocked by ADABAS in a non-standard form and are compressed, making them difficult to read. The Log will contain all updates for all files for all fields (entire records) whether all fields were changed or not. This problem is further compounded by the fact that the records on the log are ordered chronologically and are difficult to sort because they have been blocked in the non-standard form.

The ADABAS Protection Log's Before and/or After Images of updated records can be displayed as one large set of characters or hex digits. However, it is very difficult to determine which field(s) changed from such a "dump", especially when updates cause different-sized values or when fields change to/from null values. For example, examine the following Before and After Images:

BEFORE:

```
 F5  DINERS CLUB &     AMERICAN EXPRESS
 -    AMOCO            BANKERS LIFE & C
ASUALTY      BRIGHAM YOUNG D Y CATTLE
 MORGAN GUARANTY TRUST, N.Y.
```

AFTER:

```
 F5  DINERS CLUB &     AMERICAN EXPRESS
 -    AMOCO      |     BANKERS LIFE & C
ASUALTY       BRIGHAM YOUNG D Y CATTLE
  MORGAN GUARANTY TRUST, N.Y.
```

The above dump reveals some useful, readable information. Changes to alphanumeric fields can sometimes be detected visually. A hex dump, which follows, might be of more use, especially for the packed and binary numbers. Obviously it is nearly impossible to decompress the following hex data without knowing the file layout (FDT, i.e., ADACMP) and very difficult and error-prone to "sight decompress" the data even if the file layout is known.

BEFORE:

```
040186F5020CC4C9D5C5D9E240C3D3E4C203500F03060F11C1D4C5D9C9C3C1D540C5E7D7D9C5E2E2
03600F03025F0206C1D4D6C3D602400403333F025F010118C2C1D5D2C5D9E240D3C9C6C5405040C3
C1E2E4C1D3E3E8010435000F0EC2D9C9C7C8C1D440E8D6E4D5C705C402E807C3C1E3E3D3C503150F
1CD4D6D9C7C1D540C7E4C1D9C1D5E3E840E3D9E4E2E36B40D54BE84B
```

AFTER:

```
040186F5020CC4C9D5C5D9E240C3D3E4C203500F03060F11C1D4C5D9C9C3C1D540C5E7D7D9C5E2E2
03600F03025F0206C1D4D6C3D602400404444F025F010118C2C1D5D2C5D9E240D3C9C6C5405040C3
C1E3E4C1D3E3E801050135000F0EC2D9C9C7C8C1D440E8D6E4D5C705C402E807C3C1E3E3D3C50315
0F1CD4D6D9C7C1D540C7E4C1D9C1D5E3E840E3D9E4E2E36B40D54BE84B
```

AUDITRE Streamlines Audit Data

Suppose a program is written to decompress and display an updated record. This display program would be of only limited use. When there are thousands of records changed on each file, simply outputting the Before and After images of the changed records in chronological order is not sufficient for most purposes. For example, an entire printout (of several thousand pages) might have to be searched to find the Before and After images for a single updated record. To investigate all the changes to thousands of records could take hours. Further, consider a large record like this one:

```
AN = 293874628                          CUST-ACCOUNT-NUMBER
CL = 1500.00                            CREDIT-LIMIT
CS = SEWICKLEY, PA 15143                CUST-CITY-ST-ZIP
CN = JOHN DOE                           CUSTOMER-NAME
CD = 700 MAIN STREET                    CUSTOMER-ST-ADDRESS
FY = 18.00                              INT-RATE-YEARLY
OCC = 3                                 OTHER-CARDS-COUNT
OC  1 = DINERS CLUB                     OTHER-CARDS
OC  2 = AMERICAN EXPRESS                OTHER-CARDS
OC  3 = VISA                            OTHER-CARDS
OLC = 3                                 OTHER-LIMITS-COUNT
OL  1 = 2000                            OTHER-CARD-LIMIT
OL  2 = 1500                            OTHER-CARD-LIMIT
OL  3 = 1800                            OTHER-CARD-LIMIT
CO = CLERK                              CURRENT-OCCUPATION
YI = 19500                              YEARLY-INCOME
PH = 412-555-1677                       HOME-PHONE
BP = 412-555-3048                       BUSINESS-PHONE
YJ = 5                                  YEARS-AT-JOB
ED = 10/19/07                           CARD-EXPIRE-DATE
DB = 01/15/66                           DATE-OF-BIRTH
MS = S                                  MARITAL-STATUS
NC = 0                                  NUMBER-CHILDREN
DL = 288726439                          DRIV-LIC-NUMBER
DS = GA                                 DRIV-LIC-STATE
```

Such a record would appear on the Protection Log if any of its many fields were changed. An auditor would have to manually go over thousands of Before and After Images to determine which were updated by matching records and fields to one another, constantly determining whether or not the change was to an "important" field. For example:

Before		**After**
AN = 293874628		AN = 293874628
CL = 1500.00	**<- - - - - ->**	**CL = 9999.99**
CS = SEWICKLEY, PA 15143		CS = SEWICKLEY, PA 15143
CN = JOHN DOE		CN = JOHN DOE
CD = 700 MAIN STREET	**<- - ->**	**CD = 172 SCAIFFE ROAD**
FY = 18.00		FY = 18.00
OCC = 3		OCC = 3
OC 1 = DINERS CLUB		OC 1 = DINERS CLUB
OC 2 = AMERICAN EXPRESS		OC 2 = AMERICAN EXPRESS
OC 3 = VISA		OC 3 = VISA
OLC = 3		OLC = 3
OL 1 = 2000		OL 1 = 2000
OL 2 = 1500	**<- - - - - ->**	**OL 2 = 1600**
OL 3 = 1800		OL 3 = 1800
CO = CLERK		CO = CLERK
YI = 19500		YI = 19500
PH = 412-555-1677	**<- - ->**	**PH = 412-555-2805**
BP = 412-555-3048		BP = 412-555-3048
YJ = 5	**<- - - - - ->**	**YJ = 6**
ED = 10/19/07		ED = 10/19/99
DB = 01/15/66		DB = 01/15/66
MS = S	**<- - - - - ->**	**MS = M**
NC = 0		NC = 0
DL = 288726439	**<- - ->**	**DL = 502378091**
DS = GA	**<- - - - - ->**	**DS = PA**

In the example above, assume that as a credit card issuer, the only important fields to the Auditor are CREDIT-LIMIT (CL), INCOME (YI), the number of credit cards issued (OCC), and each card's limit (OLxx). The Auditor is concerned about unapproved changes to a customer's credit limit or the sudden addition of many other cards not matched by a yearly income increase. This might signify that a customer is up to something dishonest. The fact that a customer has a new driver's license or a new phone number does not really concern the Auditor.

By denoting CUST-ACCOUNT-NUMBER (AN) and CUSTOMER-NAME (CN) as key fields, AUDITRE will print a customer's name and account number whenever the customer's CREDIT-LIMIT (CL) is updated.

```
*  AN = 293874628                                CUST-ACCOUNT-NUMBER
*  CN = JOHN DOE                                 CUSTOMER-NAME
B: CL = 1500.00                                  CREDIT-LIMIT
A: CL = 9999.99                                  CREDIT-LIMIT
```

With AUDITRE, a report like the above is possible by coding:

```
AUDIT          AN*,CN*,CL,FNR=12345
```

The asterisks above indicate "key fields", in this case identifying whose credit limit changed.

The task of matching Before and After Images of the same record and determining if important fields were updated is performed entirely by AUDITRE. All that is left for the Auditor is the task of determining if the changes to the important fields warrant further investigation.

With AUDITRE, all of the updates to the important fields on the CUSTOMER-DATA file can be seen, including who changed what, when they changed it, and what they changed it to/from. Selection by date, time, file, and User-ID is possible.

AUDITRE Summarizes Update Activity

At the end of the report, AUDITRE provides a recap of the activity on the Protection Log in summary form. This would be useful, for example, if changes were authorized in the credit limit for 10 people, yet 12 were changed or 3 new credit cards were issued, but AUDITRE says 4 records were added. At a glance an Auditor would know if further investigation was warranted. A typical recap would look like this:

```
FILE:   12345           DELETES: 0          UPDATES: 24      ADDS:  0

FIELD    LONG-NAME            OCC FROM  OCC TO   UPDATES  DELETES   ADDS
AN      CUST-ACCOUNT-NO                            0        0       0
CL      CREDIT-LIMIT                              10        0       0
CS      CUST-CITY-ST-ZIP                           0        0       0
CN      CUSTOMER-NAME                              0        0       0
CD      CUSTOMER-ST-ADDRESS                        2        0       0
FY      INT-RATE-YEARLY                            0        0       0
OCC     OTHER-CARDS-COUNT                          0        0       0
OC      OTHER-CARDS              1       10        4        0       0
OLC     OTHER-LIMITS-COUNT                         0        0       0
OL      OTHER-CARD-LIMIT         1       10        4        0       0
CO      CURRENT-OCCUPATION                         0        0       0
YI      YEARLY-INCOME                              0        0       0
PH      HOME-PHONE                                 1        0       0
BP      BUSINESS-PHONE                             0        0       0
YJ      YEARS-AT-JOB                               2        0       0
ED      CARD-EXPIRE-DATE                           0        0       0
DB      DATE-OF-BIRTH                              0        0       0
MS      MARITAL-STATUS                             1        0       0
NC      NUMBER-CHILDREN                            0        0       0
DL      DRIV-LIC-NUMBER                            0        0       0
DS      DRIV-LIC-STATE                             0        0       0
```

AUDITRE is not limited to reporting or selecting from just one updated file. The user is free to select records and fields from across logically related updated files.

The user can also get summary reports of update activity by file. By coding these parameters:

```
REPORT          TYPE=SUMMARY
CONTROL         FNR
```

A summary of all updates made to the database by file would be displayed as follows:

```
   FNR          COUNT           %
     1           3827        16.8
     3          12103        53.0
     9            102         0.5
    12           6789        29.7

 *****          22821       100.0
```

Notice that the summary report above indicates that updates were made to files 1, 3, 9, and 12. If 95 updates were expected to be made to the PAYROLL-MASTER file (file 9), the fact that this summary report shows that 102 records were updated on that file would warrant a deeper investigation. AUDITRE will allow the Auditor to do this by allowing the Auditor to create additional summary reports by user, hour, etc. The Auditor might now code a summary report by user and hour, such as:

```
REPORT    TYPE=SUMMARY,
               HEADING='DEEPER INVESTIGATION OF FILE 9'
INCLUDE   FNR=9
CONTROL   UID,HOUR
```

This could produce the following output:

```
DEEPER INVESTIGATION OF FILE 9

HR       USER-ID     COUNT        %
10       PYR3            5      4.9
10       PYR1           12     11.8
10       ****           17     16.7

11       PYR1           26     25.5
11       PYR2           17     16.7
11       PYR3            4      3.9
11       ****           47     46.1

15       PYR1            3      2.9
15       PYR3           10      9.8
15       ****           13     12.7

17       PYR1            9      8.8
17       PYR2            9      8.8
17       ****           18     17.6

20       RECV            7      6.9
20       ****            7      6.9

**       ****          102    100.0
```

Now the Auditor knows that the user "RECV" (an employee in the Receiving department) manipulated PAYROLL-MASTER file records after office hours. To determine exactly what this user changed, the Auditor would code a detail report using AUDITRE to analyze all of the changes to PAYROLL-MASTER file records by Userid RECV, showing only the changed fields. If this is the first audit on this particular PLOG, the Auditor may elect to run a similar report on any old, archived Protection Logs that are available. The employee might have been changing payroll records for some time, and the Auditor would certainly want to know which records were changed, when, and from what to what.

AUDITRE Provides "After the Fact" Selective Protection Logging

Another useful function of AUDITRE is provided by the OUTPUT statement. Using this statement, the contents of Protection Logs can be "subdivided" and stored on datasets for later processing. This is of great help, for example, when the update data from a Production database normally takes up 20 reels of tape. If the user was only concerned about archiving the data for updates that occurred on a few of the files, AUDITRE could be used to select only the records for these files. This might reduce the tape volume to only 3 reels (or even 1). This can be done with parameter statements like these:

```
INCLUDE FNR=(7,13,21-24,4567)
OUTPUT

or

INCLUDE FNR=7
OUTPUT
INCLUDE FNR=13
OUTPUT
•
•
•
```

This AUDITRE function can be viewed as "after the fact", selective Protection Logging.

I.2 Uses of AUDITRE

AUDITRE can be used by many different types of organizations. For example in the petroleum industry, with its enormous databases, AUDITRE could be used to verify changes of ownership interests in oil leases. If a change occurred in a particular lease's owners, AUDITRE could be used to determine what the change was, who the new owner is (if applicable), and how much of an interest this person has. AUDITRE could also be used to verify that changes in the OWNERS file were valid, and that a dishonest programmer didn't replace the name of an owner with his own or add his name to the ownership interest of a large lease.

A magazine publisher could find just as many uses for AUDITRE. An AUDITRE report could be used to determine which subscribers' EXPIRATION-DATEs had been changed. If a particular subscriber's EXPIRATION-DATE showed that 10 years were left on that person's subscription, this might appear suspicious, since renewals for only up to 3 years in advance are accepted. AUDITRE might also show a fraud where a subscriber's NAME and ADDRESS were changed to an employee's name and address.

A stock brokerage firm would also find AUDITRE to be a significant aid in keeping all of the account data straight. Using AUDITRE, it would be simple to discover that a customer's MONEY-MARKET-BALANCE had been changed to a higher number when no deposits were made and no stock sold. AUDITRE could also be used to verify that a stock transaction for the sale of 100 shares of IBM had taken place as ordered and that 200 shares were not sold by mistake. AUDITRE could also be used to track CUSTOMER-ACCOUNTS activity to help identify the firm's best customers or employees, or to uncover evidence of "churning".

The archiving capability in AUDITRE could be useful to many departments in the same organization. For example, if the Payroll Department discovered on a routine examination of cancelled checks that an employee was regularly receiving two paychecks, archived logs could be investigated to discover how long the fraud had been taking place and if other employees were also receiving more than one check.

In a manufacturing firm, the Marketing Department could examine past customer orders and determine if the demand for certain products is cyclical and which customers purchase more of specific products than others. This information could in turn be used by the Manufacturing branch to develop production schedules.

The Purchasing Department could examine RECEIVING data for changes to expected RECEIVING-DATEs, etc. to determine if certain suppliers are regularly changing delivery dates, shipping incorrect quantities of goods, or shipping the wrong goods to the firm. Inefficient suppliers could be eliminated wherever possible.

The Finance Department might examine archived ASSETS master file data and discover that certain assets are not being depreciated, or they are not being depreciated fast enough. For example, an item that should be depreciated at an accelerated rate for tax reasons might only be getting depreciated at the normal rate. An AUDITRE report might also show that certain projects are not generating the cash flow they were expected to.

There are many uses for AUDITRE. If a question can be answered by examining changes to the database or investigated by reporting on archived data, AUDITRE can help.

I.3 Benefits of AUDITRE

AUDITRE can benefit all levels of the organization:

- ***Data Processing Management***
  - Centralization: Audit data is obtained from one source, the ADABAS Protection Log.
  - Uniformity: Auditing techniques are consistent across applications.
  - Integration: AUDITRE requires no additional report systems (e.g., COBOL, NATURAL), no specialized audit exit routines, and no specialized files.
  - Cost Effectiveness: AUDITRE reduces design, development, and implementation effort reduces CPU processing overhead, and reduces or eliminates DASD storage relative to audit information being saved.
- ***Database Administrators***
  - Simplicity: AUDITRE requires minimal time for installation and training of DBA and other users.
  - Better Design: Eliminates storage of redundant audit data within ADABAS.
  - Efficiency: I/O and CPU associated with audit processing is removed from ADABAS.
  - Recovery: AUDITRE may be used to assist in recovery of certain ADABAS data. By processing AUDITRE decompressed output data, a user-written program may be used to repair ADABAS data that had mistakenly been altered or deleted. AUDITRE can also be used to monitor changes to NATURAL Source and Object programs. These changes are reported as standard AUDITRE Before and After Image data for specified ADABAS file(s) (e. g., FNR=8, with fields LJ,LK1-9 specified in order to see NATURAL source changes). By processing AUDITRE decompressed output data for NATURAL source program fields LJ and LK as specified above, a user-written program may be used to repair NATURAL programs that had mistakenly been altered or deleted.
- ***Systems Analysts***
  - Data Validation: During development and implementation, AUDITRE can be used to validate changes to the database.
  - Audit Reporting: This easy-to-use system will satisfy the audit requirements for all applications. Multiple selection and reporting capabilities are possible in one execution of AUDITRE.
  - Responsiveness: Reports are adjustable as Auditor/User requirements change during the life of the system.

- ***Data Processing Auditors***
  - Ease of use: Only a few parameter statements are necessary, so the auditor can produce useful reports within minutes.
  - Security: Audited data is generated by ADABAS processing, not by application controlled audit logic. By allowing reports based on fields, files, etc., AUDITRE provides the auditor with clear, organized reports of database modification activity as needed for spot checks or monitoring of critical or trouble areas of the database.
  - Selectivity: Audits may be performed across logically related files on specific ADABAS fields.
  - Archival: Selected data or reports may be archived to tape or disk for later processing.
  - Compliance Testing: Because generation of audit data is removed from the control of individual applications, it is possible to perform accurate compliance testing of the database update procedures used by application software. This allows Auditors to determine if applications are complying with proper update procedures.
- ***End Users***
  - Concise Reports: Reports contain only the files and fields desired. Reports show both ADABAS long and short field names.
  - Timeliness: Reports may be quickly and easily created or modified to meet changing organizational requirements.
  - Flexibility: Reports are easy to adjust as requirements change during the application life cycle.

I.4 AUDITRE Capabilities

The function of AUDITRE is to report on ADABAS data updates, whether from NATURAL, ADABAS Direct Calls, ADABAS Native SQL, or other applications. AUDITRE can be used as the basis of a Generalized Auditing Facility at each ADABAS installation.

AUDITRE uses the ADABAS Protection Log Dataset as its primary input.

With ADABAS, there are no capabilities for selectively logging certain updates by time, file, field, etc. Either Protection Logging is on for all files, all fields, all the time, or it is off completely. In addition, there are no user-exits (as with Command Logging) within which selective logging logic could be imbedded. This "gross logging" of all updates is disadvantageous for efficiency reasons, but has the indirect advantage of enforcing auditing standards on all ADABAS applications. That is, whether desired or not in any application, auditing of the application is always possible in a standard manner as long as the DBA has Protection Logging turned on.

For the intended Backout/Regeneration function of the Protection Log, the compressed data records that ADABAS writes to its data storage blocks are also written to the Protection Log. Additionally, ADABAS writes Associator Descriptor Value Table information to the Protection Log. For Auditing purposes, the compressed data must be decompressed, and the Associator information is ignored.

AUDITRE can be used to generate multiple auditing reports in one pass of multiple ADABAS Protection Logs with selection by file and field from the compressed images on the logs. The data is decompressed and displayed or output to sequential datasets.

The Protection Log has no direct indication of which ADABAS commands (Update, Add, or Delete) caused the log records to be written. By matching corresponding Before and After Images, the AUDITRE AUDIT statement makes a determination about whether the data modification was an Update, Add, or Delete.

I.5 AUDITRE Operational Environment

AUDITRE executable code is distributed as load modules on Z/OS and as object modules, which must be link-edited on VM and VSE. AUDITRE operates in any Z/OS, VSE , or VM environment.

Installation should take less than a 1/2 hour. AUDITRE requires no zaps to ADABAS, the operating system, the teleprocessing system, or any other system software.

With the exception of the ADABAS Protection Log, AUDITRE requires no components of ADABAS or NATURAL (i.e., Utilities, PREDICT, NATURAL Security, etc.).

Refer to Section IV Installation and Operations for more information on the installation and operation of AUDITRE.

Note: Throughout this manual, references to AUDITRE imply AUDITRE7 and/or AUDITRE8.

This page intentionally left blank.

SECTION II

ADABAS PROTECTION LOG PROCESSING

This section describes the ADABAS Protection Log and lists the log fields that can be referenced in AUDITRE batch reporting parameter statements.

II.1 ADABAS Protection Log Records

AUDITRE uses the ADABAS Protection Log to produce reports. This Log contains detailed information about each database modification (Record Update, Add, and Delete). There is no ADABAS user-exit provided for intercepting and manipulating Protection Log records. If Protection Logging is requested, every Update, Delete, or Add will result in an entire Before and/or After Image of the modified records being written to the Protection Log. If the PLOG input includes spanned block records, the ADABAS parameters creating the PLOG must be defined with SRLOG=ALL to ensure the entire Before and/or After Image is included on the PLOG. The Log must be active for any ADABAS session desired to be processed by AUDITRE. There are no optional records, fields, or buffers on the Protection Log.

Multiple ADABAS sessions' Protection Logs (on disk or tape) can be processed in one AUDITRE execution. Multiple reports can be generated with one pass of these Logs. The reports can audit from the detailed Log information, identifying ADABAS files, records, and fields updated from the Before and/or After Compressed Record Images.

Execution of AUDITRE against the Protection Log is controlled by parameters with simple statements that reference the Log's "fixed fields" (DISPLAY FNR,ISN, etc.), "derived fields" (DISPLAY HOUR, DEPT, etc.), and "image fields" (AUDIT BA,BB,BC,FB,CA, etc.).

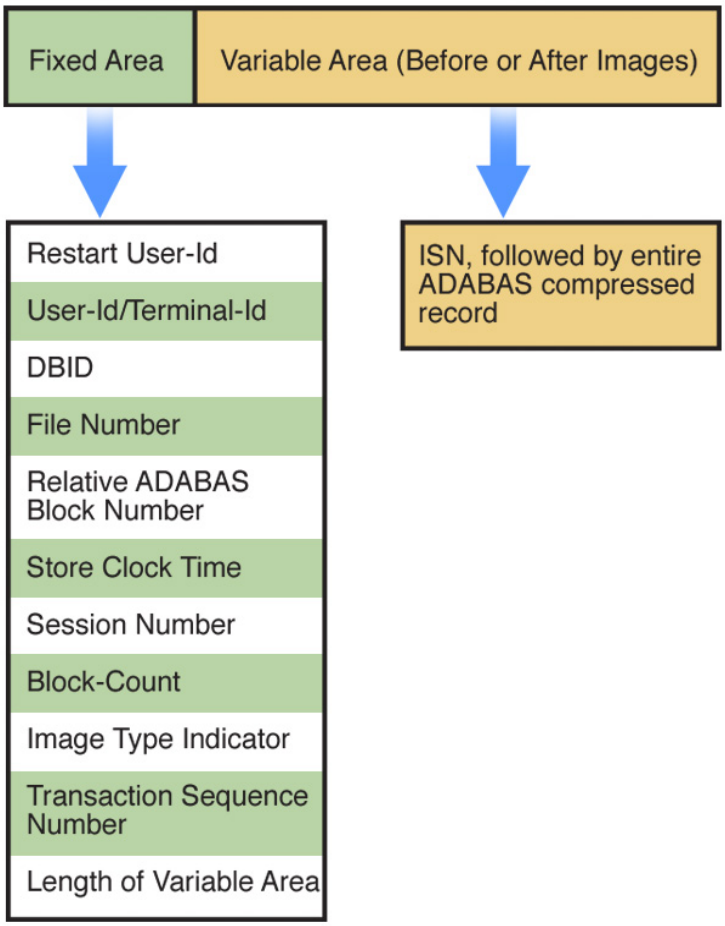
Protection Log records can be selected for particular dates, times, file numbers, ISNs, etc. It is possible to generate summarization statistics based on the "fixed and derived fields". For example, summaries for the number of Protection Log records for each file or for each hour may be generated.

II.2 ADABAS Protection Log Record Fields

The ADABAS Protection Log contains several record types. The only record type processed by AUDITRE is the Data Record Type. The header or "fixed" part of the Data Record contains several useful fields, such as File Number, User-ID, a STCK time of update (from which date and time are derived), etc. The image-type (Before or After) is indicated by a bit setting. The image portion is compressed data preceded by a length and ISN.

See Figure 2.

FIGURE 2 - PROTECTION LOG RECORD STRUCTURE

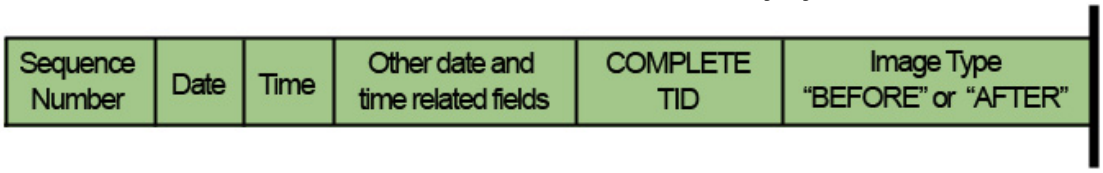


Each Protection Log Data Record that is input to AUDITRE has its fixed area converted into a record, which has a predictable, easy-to-use format. Refer to Figure 3.

FIGURE 3 - PLOG Fields Usable "as is"

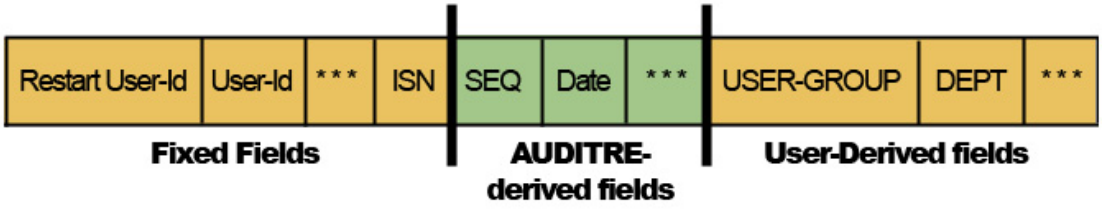
AUDITRE examines the STCK value, USER-ID, and IMAGE type indicator and creates a Sequence Number to further expand the PLOG record. Refer to Figure 4.

FIGURE 4 - PLOG Fields "Derived" Automatically by AUDITRE



The user may further "expand" the log record with FIELD statements to define derived fields, such as USER-GROUP, DEPT, etc. Refer to Figure 5.

FIGURE 5 - PLOG Fields "Derived" by User



The fixed fields and expanded or derived log fields may be referenced in various AUDITRE parameter statements.

The Before and After Image compressed fields may be referenced only in the SHOW or AUDIT statements and only by their ADABAS 2-character field names.

Figure 6 lists the fields contained in the fixed part of the log followed by the AUDITRE-derived fields showing the field names, alias names, definition, format, and length.

FIGURE 6 - EXPANDED PROTECTION LOG FIELDS

FIELD NAME	ALIAS NAMES	DEFINITION	FORMAT CHAR/ BINARY	LENGTH IN BYTES
FNR	FILE	ADABAS File Number	B	2
ISN		Internal Sequence Number of ADABAS Record	B	4
USERID8	USER-ID8 UID8	User-ID as 8 bytes (printed in character format)	C	8
USERID	USER-ID UID	User-ID as 4 bytes (printed in character format)	C	4
USERIDX	USER-IDX UIDX	User-ID as 4 bytes (printed in hex format)	C	4
RUI	RESTART-USERID	Restart User-ID	C	8
IMAGTYP	IMAGE-TYPE	"BEFORE" or "AFTER"	C	6
SESSION	SESSION-NUMBER	ADABAS Session Number	B	2
BLOCK-COUNT		BLKCNT	Block Count for use with B4 ADAREP checkpoint file list	
RABN		Relative ADABAS Block Number	B	4
TSN		Transaction Sequence Number	B	4
DBID		Database ID	B	2
LEN	RECLEN, RECORD-LENGTH	Image Record Length	B	2

This marks the end of the fields on the header or "fixed" part of the Protection Log. The fields automatically "derived" by AUDITRE from information contained on the log are listed next.

(continued on next page)

FIGURE 6 - EXPANDED PROTECTION LOG FIELDS (continued)

FIELD NAME	ALIAS NAMES	DEFINITION	FORMAT CHAR/ BINARY	LENGTH IN BYTES
SEQUENCE	SEQ	Sequence number of input record (max 6 digits printed)	B	4
SEQ7		Sequence number of input record (max 7 digits printed)	B	4
SEQ8		Sequence number of input record (max 8 digits printed)	B	4
TID	TERMINAL-ID	Terminal-ID (for COM-PLETE users)	B	2
DATE	YYDDD YY-DDD	Julian Date (Year/Day)	C	5
DATE4	YYYYDDD, YYYY-DDD	Julian Date (Year as 4 Digits/Day)	C	7
YYMMDD	YY-MM-DD	Date in Year, Month, Day Format	C	6
YYYYMMDD		Date in Year as 4 Digits, Month, Day Format	C	8
TIME		Time of Day Command entered ADABAS in hours, mins, secs.	C	6
DATETIME	DATE-TIME	YYMMDD and Time together	C	16
DATE4TIME	DATE4-TIME	YYYYMMDD and Time together	C	16
HOUR	HR	Hour of the Day (0-23)	B	1
MINUTE	MIN MI	Minute of the Hour (0-59)	B	1
DAY	DA	Day of Month (1-31)	B	1
WEEK	WK	Week of Year (1-53)	B	1
MONTH	MO	Month of Year (1-12)	B	1
WEEKDAY	WEEK-DAY	Day of Week expressed as 3 characters (SUN, MON, etc.)	C	3

(continued on next page)

FIGURE 6 - EXPANDED PROTECTION LOG FIELDS (continued)

FIELD NAME	ALIAS NAMES	DEFINITION	FORMAT CHAR/ BINARY	LENGTH IN BYTES
MONTH-NAME	MONAME	Month of Year expressed as 3 characters (JAN, FEB, etc.)	C	3
QUARTER	QU	Quarter of Year (1-4) determined by week (13 weeks per quarter except last quarter -- possible 14 weeks)	B	1
YEAR	YR	Two-digit Year	B	1
YEAR4	YR4	Four-digit Year	B	2

Note: The fields from the compressed records are referred to by their ADABAS names (AA, BB, etc.) in and only in SHOW and AUDIT statements.

All of the fields in Figure 6 that are character format will be printed in character form on all reports with the exception of USERIDX, which is printed in hex; DATE, YYMMDD, TIME, and DATE-TIME that are printed in edited form as YY-DDD, YY-MM-DD, HH:MM:SS, and YYMMDD--HH:MM:SS respectively; and the four-digit date related fields DATE4, YYYYMMDD, and DATE4-TIME which are printed in edited form as YYYY-DDD, YYYY-MM-DD, and YYYYMMDD**HHMMSS respectively.

All of the fields in Figure 6 that are numeric format will be printed as numeric edited values. Leading zeros are suppressed.

Any of the fields may be used in INCLUDE, EXCLUDE, VALUE, and Detail Report DISPLAY statements.

The format and length of values stated on INCLUDE, EXCLUDE, and VALUE statements should agree with the format and length of the stated field (e.g., "FNR=12" and "WEEKDAY=MON"). DATE should be stated as numeric characters without editing symbols (e.g., DATE=12031). YYMMDD should be stated as numeric characters without editing symbols (e.g., YYMMDD=120131). TIME should be stated as numeric characters without editing symbols (e.g., TIME=(180500-180600)). DATETIME should be stated as numeric characters with editing symbols (e.g., DATETIME=120131**18:05:00). DATE4 should be stated as numeric characters without editing symbols (e.g., DATE=2012031). YYYYMMDD should be stated as numeric characters without editing symbols (e.g., YYYYMMDD=20120131). DATE4TIME should be stated as numeric characters with editing symbols (e.g., DATE4TIME=20120131**180500).

The DATETIME field represents a combination of the date and time-of-day into one field. Although separate expressions of date and time-of-day are meaningful in a non-technical sense, efficiency in data processing requires a point in time to be represented as year/month/day/hour/minute/second. This allows for sorting and identifying specific points in time, such as earliest and latest points in time.

DATETIME is a combination of the YYMMDD field and the TIME field, which are both normally 8 characters. In order to fit these two fields into the AUDITRE internal limit of 16 characters per field, the YYMMDD portion is represented without editing symbols, while the TIME portion is represented with colons as editing symbols. Separating the two portions are two asterisks. Therefore, a DATETIME example is "120131**12:59:59".

DATE4TIME is a combination of the YYYYMMDD field and the TIME field, which are both normally 8 characters. In order to fit these two fields into the AUDITRE internal limit of 16 characters per field, the TIME portion is represented without editing symbols. Separating the two portions are two asterisks. Therefore, a DATE4TIME example is "20120131**125959".

In AUDITRE, the earliest point in time for input log records can be identified on a Summary Report by stating MINIMUM DATETIME, and the latest point in time by stating MAXIMUM DATETIME.

One additional "field" is available for printing via the Detail Report DISPLAY statement:

- IMAGE, which prints in character and hex formats.

Additionally, fields defined by the FIELD parameter statement can be used in subsequent statements.

II.3 Sample Protection Log SHOW Report

Report Headings:

```
SHOW DATA - FILE 1237 AUDITRE 4.0 MON 12-01-31 19:23:51 PAGE 1
```

Fixed and Derived Fields:

```
YY-MM-DD            TIME      FNR    ISN        USERIDX      IMAGTYP

12-01-31            12:27:53     1,237     1,761,847        00001417            BEFORE
```

Decompressed image fields with short and long names:

```
BA=DAVENPORT                                  LAST-NAME
BB=CHARLES                                    FIRST-NAME
BC=M                                          INITIAL
FB=50000                                      SALARY

12-01-31            12:27:53     1,237     1,761,847        00001417             AFTER

BA=DAVENPORT                                  LAST-NAME
BB=CHARLES                                    FIRST-NAME
BC=M                                          INITIAL
FB=52000                                      SALARY

12-01-31            12:28:09     1,237     2,141,902        00001417            BEFORE

BA=TRUMBULL                                   LAST-NAME
BB=JOHN                                       FIRST-NAME
BC=C                                          INITIAL
FB=67000                                      SALARY

12-01-31            12:28:09     1,237     2,141,902        00001417             AFTER

BA=TRUMBULL                                   LAST-NAME
BB=JOHN                                       FIRST-NAME
BC=C                                          INITIAL
FB=75000                                      SALARY
```

This is a SHOW type of report specified as:

```
REPORT          TYPE=DETAIL,HEADING='SHOW DATA - FILE 1237'
DISPLAY         YYMMDD,TIME,FNR,ISN,USERIDX,IMAGTYP
SHOW            BA,BB,BC,FB,FNR=1237
```

Similar reports can be generated for other files all in one PLOG pass. Related files can have their fields shown all on one report.

Numerous image fields can be shown. Each image field is shown on a separate line of the report to allow for a possible lengthy display followed by the long name.

Many more fields can be listed than those shown in the sample report. Note that multiple occurring fields (MUs and PEs) will use a lot of report space when large numbers of occurrences are requested.

Examination of a lengthy SHOW report to locate update activity can be time consuming and error prone. SHOW reports do have value as "dumps of PLOG data in readable form". The SHOW statement and SHOW reports are more fully described in **Section III.2.7 SHOW** and in **Appendix C Sample Protection Log**.

II.4 Sample Protection Log AUDIT Report

Compared to a SHOW report, an AUDIT type of report produces a smaller, more concise detail report, which clearly shows all update activity desired.

An AUDIT report can be thought of as two SHOW reports, one for Before Images and one for After Images, held next to each other and compared line for line with all duplicate lines erased. What remains is update information. After viewing the previous sample report, it is obvious (on one simple report page) that two people's salaries were updated, but when a SHOW report for one file involves hundreds or thousands of pages, the data modifications are best exposed through AUDIT reports. Thousands of pages can often be reduced to a handful.

When reported via an AUDIT report, the previously shown updates appear as follows:

Report Headings:

```
AUDIT - FILE 1237  AUDITRE 5.0 MON 12-01-31 19:23:51 PAGE 1
```

Fixed and Derived Fields:

```
YY-MM-DD             TIME          FNR       ISN             USERIDX

12-01-31             12:27:53      1,237      1,761,847       00001417
```

Decompressed Key and Changed Image Fields with Short and Long Names:

```
*    BA=DAVENPORT                                          LAST-NAME
*    BB=CHARLES                                       FIRST-NAME
 B:  FB=50000                                                    SALARY
 A:  FB=52000                                             SALARY

12-01-31             12:28:09       1,237     2,141,902       00001417

*    BA=TRUMBULL                                        LAST-NAME
*    BB=JOHN                                          FIRST-NAME
 B:  FB=67000                                             SALARY
 A:  FB=75000                                             SALARY
```

This is an AUDIT type of report specified as:

```
REPORT         TYPE=DETAIL,HEADING='AUDIT - FILE 1237'
DISPLAY        YYMMDD,TIME,FNR,ISN,USERIDX
AUDIT          BA*,BB*,FB,FNR=1237
```

With these parameters, the user is examining FB (Salary Field) changes. The AUDIT statement instructs AUDITRE to decompress and compare BA, BB, and FB fields. If any of these fields' values have changed, the Before and After values (note the B: and A: in the previous example) are to be exposed. Additionally, the record is to be identified by printing values of "key fields" (BA and BB). Key fields are indicated by an asterisk in the AUDIT statement field list and in the first column of the reports' field displays.

This report answers the questions:

- whose salaries were changed
- from what value, to what value
- when
- by whom (User-ID)

As in SHOW Reports, multiple AUDIT Reports can also be generated all in one PLOG pass. Related files can have fields audited all on one report. Numerous image fields can be specified for decompression and comparison. No matter how many fields are stated, even MUs and PEs, reports are usually small because only changed fields are identified. However, unnecessarily listing excessive numbers of fields or large occurrence ranges will result in a greater CPU utilization.

In the cases of record Adds (After Image only) and Deletes (Before Image only), AUDITRE presents options of printing all of the fields listed or printing only the key (asterisk) fields. With either option, null valued fields are not printed.

In the case of record Updates, AUDITRE presents two options for records that were updated but for which no fields are listed for AUDIT. AUDITRE may be instructed to identify the key (asterisk) fields or to print nothing about the update, since none of the important (listed) fields were updated.

At the completion of each AUDIT report, tallies of Updates, Adds, and Deletes are shown for each file and field in each report. This valuable summarization may reveal, for example, seven salary updates rather than six that were approved.

The AUDIT statement and AUDIT reports are more fully described in **Section III.2.8 AUDIT** and in **Appendix C Sample Protection Log**.

SECTION III

LOG ANALYSIS PARAMETER STATEMENTS

III.1 Parameter Statements and Types

For AUDITRE Protection Log processing, there are several types of parameter statements with op-codes as shown below.

Input Definition

INPUT	Defines the input and other "global" information.
FIELD VALUE	Specifies any derived fields.

Report Definition

REPORT	Defines the report type, heading lines, format, etc.

Record Selection

INCLUDE EXCLUDE	Specifies which input records should be included in or excluded from the report.

Detail Report Field Specification

DISPLAY	Specifies which fields from the Protection Log "fixed or derived fields" should be printed on the report.
SHOW	Specifies which fields from the Protection Log Before and/or After Images are to be decompressed and shown on the report.
AUDIT	Specifies which fields from the Protection Log Before and After Images are to be decompressed, compared for change, and reported.

Summary Report Specification

CONTROL	Specifies which field(s) to use for control breaks.

Sequential Dataset Output Definition

OUTPUT	Defines a Sequential Dataset to contain either output copies of Protection Log compressed records in the same form as input, decompressed data from the Protection Log in a "flat file" form, or all data shown on a summary report.

III.2 Parameter Statement Syntax and Description

For general parameter syntax rules, refer to **Appendix A General Parameter Rules**.

The Parameter Statement Syntax and a detailed description of each parameter statement with examples follow.

III.2.1 INPUT

The INPUT Statement defines the input and sets certain "global" parameters for the run. Only one INPUT Statement is permitted.

SYNTAX

```
INPUT   LOGTYPE          =   PROTECTION

        LIMIT            =   { numeric-value
                               99999999          }

        CLOCK-FACTOR     =   { numeric-value
                               0                 }

        TAPES            =   { numeric-value                 (VSE and
                               0                 }           VM only)

        VMBLK            =   { YES
                               NO  }                         (VM only)

        SYSNO            =   { 010
                               012 }                         (VSE only)

        REWIND           =   { YES
                               NO  }                         (VSE only)

        STARTDATE        =   { yymmdd
                               000000  }

        STARTDATE4       =   { yyyymmdd
                               00000000  }

        STARTTIME        =   { hhmmss
                               000000  }


        STOPDATE         =   { yymmdd
                               999999  }

        STOPDATE4        =   { yyyymmdd
                               99999999  }

        STOPTIME         =   { hhmmss
                               999999  }
```

LOGTYPE: This defines the input to be the Protection Log. AUDITRE is pre-coded to know the format of the log.

LIMIT: This specifies the maximum number of input records to be processed.

CLOCK-FACTOR: This specifies the number of hours difference between the times contained in the log and the real time. For example, several installations for one company may IPL all systems with Greenwich Mean Time as the base clock for synchronization purposes. The log is from an ADABAS session run in Central Standard Time, 6 hours different from Greenwich Mean Time. The Clock-Factor is 6, which will cause log times to be reported based upon Central Standard Time. The Clock-Factor can be a positive or negative integer value.

TAPES: This specifies the number of input tapes to be processed (for VSE or VM). If specified as 0, AUDITRE will assume some unknown multiple number of tapes (files) are to be processed. Upon end-of-file, AUDITRE will ask the operator if more tapes are to be processed. If specified as non-zero, AUDITRE will assume input of the specified number of tapes (files), stopping only when all are processed. (For VM, "TAPES=1" should be specified when processing logs from a disk file.)

VMBLK: In certain installations, VM inserts an extra record length in front of the log record being input. "NO" (the default) makes no adjustment for this extra four bytes. "YES" makes a four byte adjustment to skip over this extra length.

SYSNO: For VSE , input can be from disk (SYSNO=012) rather than tape (SYSNO=010, the default). For OS and VM, this is accomplished through JCL.

Refer to **Section IV.7.5 VSE**.

REWIND: For VSE, the input tape(s) may be rewound (REWIND=YES) upon end of file (for another tape to be mounted) or left in position on the tape drive (REWIND=NO) for multiple files on one reel. This parameter is ignored for non-VSE and for VSE SYSNO=012.

STARTDATE: The sequence of PLOG records and the sequence of multiple tapes or volumes of records input to AUDITRE must be in chronological order as produced by ADABAS. Installations must guarantee date/time ordered input through operations procedures, JCL, generation datasets, tape management systems, etc. Because the input date is ordered, installations may take advantage of this parameter, STARTDATE, which along with STARTTIME specifies a specific starting date/time.

STARTDATE4: Refer to STARTDATE above. As an option a four-digit ending date can be specified. It is possible to state STARTDATE4=20120131 or STARTDATE=120131, for example. These are both interpreted as starting PLOG processing on January 31, 2012.

STARTTIME: See STARTDATE above.

STOPDATE: The sequence of PLOG records and the sequence of multiple tapes or volumes of records input to AUDITRE must be in chronological order as

produced by ADABAS. Installations must guarantee date/time ordered input through operations procedures, JCL, generation datasets, tape management systems, etc. Because the input date is ordered, installations may take advantage of this parameter, STOPDATE, which along with STOPTIME specifies an ending date/time. When this date/time is exceeded, AUDITRE inputting will cease.

STOPDATE4: Refer to STOPDATE above. As an option a four-digit ending date can be specified. It is possible to state STOPDATE4=20120131 or STOPDATE=120131, for example. These are both interpreted as stopping PLOG processing on January 31 of 2012.

STOPTIME: See STOPDATE above.

EXAMPLE:

```
INPUT   LOGTYPE=PROTECTION,TAPES=3,STOPDATE=120131,
                STOPTIME=092253
```

All Protection Log records are to be processed up to and including the stop date/time. This is a VSE or VM execution against exactly 3 tape files, but the processing may be terminated if a record has a date and time exceeding the indicated STOPDATE and STOPTIME.

III.2.2 FIELD

The FIELD statement defines fields in addition to those fields that AUDITRE assumes to be on the Protection Log. The FIELD statement is used in conjunction with the VALUE statements to "derive" field values.

SYNTAX

```
FIELD: NAME                    =  Character-string

                FORMAT         =  {  C
                                     B
                                     H   }

                LENGTH         =  numeric-value

                DECIMALS       =  {  numeric-value
                                     0                  }
```

NAME: Up to 8 alphabetic, hyphen, and numeric characters to give the field a name that can be used in subsequent parameter statements. For example, different departments, divisions, sections, or projects can be defined; certain file-groups can be identified for reporting purposes; etc.

FORMAT:
- C = Character format field
- B = Binary numeric field
- H = Character format field (prints in hex)

LENGTH: For 'C' or 'H' format, one to 16 bytes of length. For 'B' format, one, two, or four bytes of length.

DECIMALS: For 'B' format, one to five decimal positions for printing the field.

EXAMPLES

```
FIELD   NAME=DEPT,LENGTH=10,FORMAT=C
```

A field named 'DEPT' is to be considered as a log field. DEPT is a 10 character field. The value of DEPT will be "derived" via subsequent VALUE statements.

```
FIELD   NAME=INTERVAL,LENGTH=2,FORMAT=B,DECIMALS=2
```

A field named 'INTERVAL' is to be considered as a log field. INTERVAL is 2 bytes binary with 2 decimal positions. The value of INTERVAL will be "derived" via subsequent VALUE statements.

```
FIELD   NAME=DBNAME,LENGTH=4,FORMAT=C
```

A field named 'DBNAME' is to be considered as a log field. DBNAME is a 4 character field. The value of DBNAME will be "derived" via subsequent VALUE statements.

III.2.3 VALUE

The VALUE statement defines a value that the preceding field (FIELD statement) may have and the criteria for setting that value.

SYNTAX

```
VALUE     { character-string
            numeric-value        }   relation-expression[,relation-expression]...
```

If the preceding field is format 'C' or 'H', a character string of any length up to the length specified for the preceding field must be specified.

If the preceding field is format 'B', a numeric value (decimal) capable of fitting into the byte length defined for the preceding field must be specified.

The relation-expressions state the criteria for setting the value of the preceding field to the character string or numeric value stated. Each relation-expression is made up of a field, a relation-operator, and a value-list:

- field: a field in the input record, such as FNR, ISN, etc., or a previously defined derived field, such as DEPT.

- relation-operator:
  - `=` (equal)
  - `>` (greater than)
  - `<` (less than)
  - `¬` (not equal)

- value-list: a list of one or more values. If more than one, the values must be enclosed in parentheses and separated by commas. Value ranges are stated as value-value. The value format should agree with the input field format (e.g., RESTART-USERID is alphanumeric, FNR is numeric).

Multiple values following an equal operator imply logical OR relationships. Multiple values following a not equal operator imply logical AND relationships.

Multiple relation-expressions imply logical AND relationships.

EXAMPLES (See the previous FIELD examples.)

```
FIELD          NAME=DEPT,LENGTH=10,FORMAT=C
VALUE          ENG,RUI=(A400AABA-A4999999)
VALUE          MARKETING,RUI=(M000AAAA-M9999999),
                       FNR=(30-33,37,1746,39)
VALUE          DBA,FNR=387
VALUE          MISC
```

Set the value of the derived field 'DEPT' to 'ENG' if the RESTART-USERID on the log is within the indicated range of values. If not, set the value to 'MARKETING' if the RESTART-USERID on the log is within the indicated range of values AND the file-number on the log is within the indicated range. If not, set the value to 'DBA' if the file number is 387. If not, set the value to 'MISC'.

```
FIELD          NAME=INTERVAL,LENGTH=2,FORMAT=B,
                      DECIMALS=2
VALUE          .00,MINUTE=(0-14)
VALUE          .25,MINUTE=(15-29)
VALUE          .50,MINUTE=(30-44)
VALUE          .75                                  MINUTE MUST BE 45-59
```

Set the value of the derived field 'INTERVAL' to a decimal value to indicate quarter hour intervals. This will be useful for breaking down statistics by hour and interval of hour.

```
FIELD          NAME=DBNAME,LENGTH=4,FORMAT=C
VALUE          PROD
```

Set the value of the derived field 'DBNAME' to 'PROD' to indicate that the statistics will be for the production database.

The selection criteria is processed in order from top to bottom, so when coding long lists of VALUE statements, the parameter statement order can affect AUDITRE performance.

III.2.4 REPORT

The REPORT statement defines the type of report to be generated, along with heading line(s) and report format.

SYNTAX

```
REPORT    TYPE        =  {  DETAIL
                            SUMMARY      }

          HEADING     =  'up to 60 characters'

          HEADING2    =  'up to 60 characters'

          LIMIT       =  {  99999999
                            numeric-value      }

          LINE-SIZE   =  {  133
                            numeric-value      }

          PAGE-SIZE   =  {  55
                            numeric-value      }
```

As many as 98 distinct detail reports may be specified (5 for VSE). An unlimited (within storage constraints) number of summary reports may be specified.

TYPE: Defines whether the report is to be a detail report or a summary report.

HEADING and HEADING2: HEADING specifies the first heading line that will appear at the top of the report. HEADING2 specifies the optional second heading line.

LIMIT: Specifies the maximum number of non-heading lines to be printed on the report.

LINE-SIZE: Defines the report width or line size, including the carriage control character.

PAGE-SIZE: Defines the number of non-heading lines to print per page.

EXAMPLES:

```
REPORT  TYPE=DETAIL,HEADING='AUDIT OIL REVENUE',
          LIMIT=2000
```

A detail report with the indicated heading should be printed. The user only wants to view the first 2000 lines of the report, possibly to see if the report prints as desired or to limit the report lines while outputting all the audit data to an output data set. The report will be the standard 133 characters by 55 lines per page.

```
REPORT  TYPE=SUMMARY,HEADING='MARKETING DIVISION UPDATES',
          HEADING2='FILE 537 AND 543 ONLY'
```

A summary report with the indicated headings should be printed. The report will be the standard 133 characters by 55 lines per page.

III.2.5 INCLUDE/EXCLUDE

The INCLUDE and EXCLUDE statements specify the selection criteria for input records to be processed in the preceding specified REPORT. If no INCLUDE or EXCLUDE statements are specified, all input records are processed in this report.

SYNTAX

{ INCLUDE
EXCLUDE } relation-expression[,relation-expression]...

Each relation-expression is made up of a field, a relation-operator, and a value-list.

- field: a field in the input record, (e.g., FNR, ISN, USERID), an AUDITRE-derived field (e.g., HR, DATE, etc.), or a previously defined derived field, such as DEPT.

- relation-operator:
  - = (equal)
  - > (greater than)
  - < (less than)
  - ¬ (not equal)

- value-list: a list of one or more values. If more than one, the values must be enclosed in parentheses and separated by commas. Value ranges are stated as value-value. The value format should agree with the input field format (e.g., RESTART-USERID is alphanumeric, FNR is numeric).

Multiple values following an equal operator imply logical OR relationships. Multiple values following a not equal operator imply logical AND relationships.

Multiple relation-expressions imply logical AND relationships.

If the input record matches the INCLUDE/EXCLUDE selection criteria, it is immediately included in or excluded from further processing for the report.

EXAMPLES:

```
INCLUDE        RUI=(C4103X27,C402AAAA-C4029999),FNR=8
```

Input records will be included in the report only if the value of the RESTART-USERID field is C4103X27, *OR* C402AAAA through C4029999, *AND* the file-number is 8. Otherwise, the records are *excluded*.

```
EXCLUDE        ISN=(1000-2000)
INCLUDE        FNR=(1-50,401-450)
EXCLUDE        RUI=ENGTEST1,FNR=18347
```

Input records are excluded from the report if the ISN is 1000 to 2000. Further, input records not previously excluded are included if the file number is in the range 1-50 *OR* the range 401-450. Further, input records not previously included are excluded if the RESTART-USERID is ENGTEST1 *AND* the file-number is 18347. Otherwise, the records are *included*.

III.2.6 DISPLAY

The DISPLAY statement indicates the fixed and/or derived input fields to be printed on the detail report defined by the preceding REPORT statement.

SYNTAX (Option 1)

```
DISPLAY     FIELD [,FIELD]...
```

The fields will be displayed in the order that they are indicated. The format, length, and field heading are pre-defined within AUDITRE.

EXAMPLE

```
DISPLAY     SEQUENCE,DATE,TIME,FNR,ISN,
                  USERID,RUI,IMAGTYP,PROJECT
```

The listed fixed fields from the Protection Log (USERID, FNR, ISN, RUI), AUDITRE-derived fields (SEQUENCE, DATE, TIME, IMAGTYP), and the user-derived field (PROJECT) will be printed on the detail report one line per record as follows:

```
SEQ   YY-DDD HH-MM-SS   FNR     ISN    USERID  RUI      IMAGTYP  PROJECT
•
•
•
  20  12-031 14:19:12    343       5   A3P3  BOOKREV3  BEFORE   ANP13
  21  12-031 14:19:12    343       5   A3P3  BOOKREV3  AFTER    ANP13
  28  12-031 14:19:13    786    1234   B107  PUBL43    BEFORE   B41X5
  29  12-031 14:19:13    786    1234   B107  PUBL43    AFTER    B41X5
•
•
•
```

SYNTAX (Option 2)

```
DISPLAY IMAGE
```

The Before and/or After Images can be displayed as one large field in character and hex. It would be very difficult to determine which field(s) changed from such a "dump", especially when updates cause different-sized values or when fields change to/from null values.

```
BEFORE IMAGE , LENGTH = 148   DISPLAYED IN   40  CHARACTER SEGMENTS

 F5  DINERS CLUB &     AMERICAN EXPRESS
 -   ┐  AMOCO        ┐   BANKERS LIFE & C
ASUALTY      BRIGHAM YOUNG D Y CATTLE
 MORGAN GUARANTY TRUST, N.Y.


AFTER  IMAGE , LENGTH = 148   DISPLAYED IN   40  CHARACTER SEGMENTS

  F5  DINERS CLUB &     AMERICAN EXPRESS
 -   ┐  AMOCO     | ┐   BANKERS LIFE & C
ASUALTY      BRIGHAM YOUNG D Y CATTLE
 MORGAN GUARANTY TRUST, N.Y.
```

The "character" dump above shows some useful, readable information. Changes to alphanumeric fields can sometimes be detected.

The "hex" dump, which follows, appears on the printout to the right of the character information shown on the previous page. It is nearly impossible to decompress the hex data without knowing the file layout (FDT) (i.e., ADACMP) and very difficult and error-prone to "sight decompress" the data even if the file layout is known.

Before:

```
040186F5020CC4C9D5C5D9E240C3D3E4C203500F03060F11C1D4C5D9C9C3C1D540C5E7D7D9C5E2E2
03600F03025F0206C1D4D6C3D602400403333F025F010118C2C1D5D2C5D9E240D3C9C6C5405040C3
C1E2E4C1D3E3E8010435000F0EC2D9C9C7C8C1D440E8D6E4D5C705C402E807C3C1E3E3D3C503150F
1CD4D6D9C7C1D540C7E4C1D9C1D5E3E840E3D9E4E2E36B40D54BE84B
```

After:

```
040186F5020CC4C9D5C5D9E240C3D3E4C203500F03060F11C1D4C5D9C9C3C1D540C5E7D7D9C5E2E2
03600F03025F0206C1D4D6C3D602400404444F025F010118C2C1D5D2C5D9E240D3C9C6C5405040C3
C1E3E4C1D3E3E8010435000F0EC2D9C9C7C8C1D440E8D6E4D5C705C402E807C3C1E3E3D3C503150F
1CD4D6D9C7C1D540C7E4C1D9C1D5E3E840E3D9E4E2E36B40D54BE84B
```

III.2.7 SHOW

The SHOW statement lists the Protection Log Compressed Data Record Fields to be printed on the associated detail report. If an OUTPUT statement is also specified for this report, the SHOW statement causes each shown field to be placed in order in one record on the output sequential dataset. Refer to **Appendix B Output Dataset Formats** for the exact OUTPUT format.

Because the data records in the Before and After Images on the Protection Log are compressed ADABAS records, the fields to be shown must be decompressed to be printed and/or output. In order to decompress the data, AUDITRE needs to know the file description (FDT). This is supplied to AUDITRE via control cards similar to compression (ADACMP) utility card images for each file having fields to be shown. An example is shown later in this subsection.

SYNTAX

```
SHOW      {ALL
           ADABAS-FIELD-LIST}...,

FNR       =      file number
```

ADABAS-FIELD-LIST: This is a list of ADABAS 2-character field names separated by commas.

The elementary ADABAS fields will be printed and/or output in the order they are stated. The printed fields will be identified by their ADABAS field name and column 46-72 data from the ADACMP card images (usually long names). The print length and format will be determined by AUDITRE. Alphanumeric fields will print in character format with a maximum of 85 characters per line. Up to three lines may be used to print large alphanumeric fields. Unpacked and packed fields (up to eight bytes) will be printed as whole decimal numbers with a possible minus sign. Binary fields will have their last seven bytes printed in hex and in decimal.

Each SHOW statement's ADABAS-FIELD-LIST is limited to 100 ADABAS fields. With multiple SHOW statements, this limit is easily avoided.

ALL: This is for use in those cases where listing all elementary fields would be burdensome. In terms of efficiency, ALL is more expensive than the selection of a few important fields to show.

FNR: This states the ADABAS file number. This parameter must be stated, otherwise file zero will be assumed, ending the run as a JCL error for lack of ADACMP card images for file zero.

When more than one file's data is being shown, two or more reports can be generated, one for each file. If two or more files have related data (Personnel and Finance files), then they can be included in one report, but they must be stated as two or more separate SHOW statements. When multiple files are referenced in one report, care must be taken to include all appropriate files (INCLUDE statement) for processing.

When a file has several record types (views), each view can be shown, preferably in separate reports referencing the same file, different fields.

To process the examples shown below, the Compress (ADACMP) file definition card images for file 1 (Personnel File in the examples) must be supplied in the AUDITRE execution JCL. A sample ADACMP for file 1 follows:

FIGURE 7 – SAMPLE ADACMP

```
01,AA,008,B,DE                                  PERSONNEL-NUMBER
01,BA,020,A,NU,DE                               LAST-NAME
01,BB,015,A,NU,DE                               FIRST-NAME
01,BC,001,A,FI                                  INITIAL
01,CA,001,A,NU,DE                               SEX
01,CB,002,U,NU,DE                               AGE
01,CC,010,A,NU,DE                               FAMILY-STATUS
01,CD,002,U,NU,DE                               DEPENDENTS
01,DA,005,U,NU                                  NUMBER
01,DB,020,A,NU,DE                               STREET
01,DC,015,A,NU,DE                               CITY
01,DD,002,A,NU,DE                               STATE
01,DE,005,U,NU,DE                               ZIP-CODE
01,DF,008,A,NU,DE                               PHONE-NUMBER
01,FA,020,A,NU,DE                               JOB
01,FB,006,U,NU,DE                               SALARY
01,FC,006,U,NU                                  COMMISSION
01,GA,002,U,NU                                  YEARS-OF-EDUCATION
01,HA,002,U,NU                                  YEARS-WITH-COMPANY
01,IA,002,U,NU                                  VACATION-DAYS
01,KA,002,U,NU                                  SICK-DAYS
01,LA,030,A,NU,DE                               HOBBY
```

EXAMPLE - ELEMENTARY FIELDS:

```
       SHOW          BA,BB,BC,AA,CA,CB,FB,DA,DB,DC,FNR=1
```

The listed fields from file 1 will be printed in the order shown one line per field, as follows:

```
BA=DAVENPORT                                    LAST-NAME
BB=ANN                                          FIRST-NAME
BC=P                                            INITIAL
AA=  HEX 00000000001C4B     DEC 7243            PERSONNEL-NUMBER
CA=F                                            SEX
CB=38                                           AGE
FB=048000                                       SALARY
DA=00126                                        STREET-NUMBER
DB=DRURY LANE                                   STREET
DC=CANOGA PARK                                  CITY
```

EXAMPLE - SHOW ALL:

```
SHOW           ALL,FNR=1
```

All elementary fields from file 1 will be printed in the order they appear in the file, as follows:

```
AA=  HEX 00000000001C4B       DEC 7243              PERSONNEL-NUMBER
BA=DAVENPORT                                        LAST-NAME
BB=ANN                                              FIRST-NAME
BC=P                                                INITIAL
CA=F                                                SEX
CB=38                                               AGE
CC=MARRIED                                          FAMILY-STATUS
CD=2                                                DEPENDENTS
DA=00126                                            STREET-NUMBER
DB=DRURY LANE                                       STREET
DC=CANOGA PARK                                      CITY
DD=CA                                               STATE
DE=91304                                            ZIP
DF=                                                 PHONE-NUMBER
FA=PROG.MGR.                                        JOB
FB=048000                                           SALARY
FC=0                                                COMMISSION
GA=14                                               YEARS-OF-EDUCATION
HA=7                                                YEARS-WITH-COMPANY
IA=13                                               VACATION-DAYS
KA=3                                                SICK DAYS
LA=RACQUET SPORTS                                   HOBBY
```

Assume the FIGURE 7 SAMPLE ADACMP for file 1 also contains:

```
01,XY,10,A,MU,NU                                    CHILDREN-NAMES
01,PP,PE                                            SALES-INFO
02,PQ,10,A,NU                                       CUST-ID
02,PR,5,P,NU                                        ITEMS-SOLD
02,PS,7,U,NU                                        DOLLARS-SOLD
02,PT,20,A,NU,MU                                    CUST-PERSONNEL
```

Note that this example now contains an MU field, a PE Group, and fields within the PE, including an MU field within the PE.

EXAMPLE - MU AND PE FIELDS:

MU and PE fields must be specified as occurrence numbers or ranges directly after the field name. For example: XY7-13,PQ12

This will print:

```
XY  7=value
XY  8=value
XY  9=value
XY 10=value
XY 11=value
XY 12=value
XY 13=value
PQ 12=value
```

All occurrences are printed one per line.

EXAMPLE - MU AND PE FIELDS:

MU and PE count fields must be stated as a field followed by 'C'. A count value will display as 3 bytes when ADABAS 7 PLOG is input and 5 bytes when ADABAS 8 PLOG is input. For example: XYC, XY7-13, PQC...

This will print:

```
XYC = 9
XY  7=value
XY  8=value
XY  9=value
XY 10=null value
XY 11=null value
XY 12=null value
XY 13=null value
PQC=7
```

When a count field, such as XYC=9, indicates fewer actual occurrences than what are stated for print, the remaining occurrences (10-13 above) SHOW as the null value (zeros or blanks as appropriate).

EXAMPLE - MU WITHIN PE FIELD:

MU within PE fields must be specified by using a # character. For example: PT3-15#4-7 means PT3, MU occurrence number 4, 5, 6, and 7, followed by PT4, MU occurrences number 4, 5, 6, and 7, etc.

This will print:

```
PT  3#  4=value
PT  3#  5=value
PT  3#  6=value
PT  3#  7=value
.
.
.
repeated for PT  4, 5, 6, ... 15
```

EXAMPLE - MU WITHIN PE COUNT FIELD:

An MU within PE count field must be specified as a field followed by the occurrence number and 'C'. For example: PT7C

This will print:

```
PT  7C=6
```

There are 6 occurrences of the MU field PT within the 7th occurrence of its PE group.

Each included record's Before and/or After Image will be printed as demonstrated above. If the data is also to be OUTPUT, each field is output according to its FDT (ADACMP) definition length and format. Count fields always are output as one binary byte when ADABAS 7 PLOG is input and two binary bytes when ADABAS 8 PLOG is input. The exact format of the output record is defined in **Appendix B Output Dataset Formats**.

EXAMPLE OF ELEMENTARY, MU, PE, AND MU WITHIN PE FIELDS:

```
SHOW    BA,BB,BC,XYC,XY3-5,PQC,PQ2-4,PT1-3#2-4,PT2C,FNR=1
```

The listed fields from file 1 will be printed in the order shown one line per field and occurrence, as follows:

```
BA=DAVENPORT
BB=ANN
BC=P
XYC=  4
XY  3=ROBERT
XY  4=MARY
XY  5=
PQC=  3
PQ  2=17346AB
PQ  3=12907DP
PQ  4=
PT   1#   2=PETER
PT   1#   3=ROBIN
PT   1#   4=PAUL
PT   2#   2=MARTHA
PT   2#   3=CHRIS
PT   2#   4=
PT   3#   2=JOSEPH
PT   3#   3=EMILIE
PT   3#   4=
PT  2C=  2
```

```
LAST-NAME
FIRST-NAME
      INITIAL
CHILDREN-NAMES
CHILDREN-NAMES
      CHILDREN-NAMES
CHILDREN-NAMES
CUST-ID
CUST-ID
CUST-ID
CUST-ID
      CUST-PERSONNEL
      CUST-PERSONNEL
      CUST-PERSONNEL
CUST-PERSONNEL
      CUST-PERSONNEL
      CUST-PERSONNEL
CUST-PERSONNEL
CUST-PERSONNEL
      CUST-PERSONNEL
      CUST-PERSONNEL
```

III.2.8 AUDIT

The AUDIT statement lists the Protection Log Compressed Data Record Fields to be decompressed, compared for change, and possibly printed on the associated detail report. If an OUTPUT statement is also specified for this report, the AUDIT statement causes each AUDIT field to be placed in order in one record on the output sequential dataset. Refer to **Appendix B Output Dataset Formats** for the exact OUTPUT format.

Because the data records in the Before and After Images on the Protection Log are compressed ADABAS records, the fields to be AUDITed must be decompressed to be printed and/or output. In order to decompress the data, AUDITRE needs to know the file description (FDT). This is supplied to AUDITRE via control cards similar to compression (ADACMP) utility card images for each file having fields to be AUDITed (example later in this subsection).

SYNTAX

```
AUDIT     {ALL
           ADABAS-FIELD-LIST}...,

              FNR      =  file number

              ADD      =  {LIST
                           *       }

              UPDATE   =  {NOTHING
                           *       }

              DELETE   =  {LIST
                           *       }
```

ADABAS-FIELD-LIST: This is a list of ADABAS 2-character field names separated by commas.

The elementary ADABAS fields will be decompressed, compared (Before to After), printed, and/or output in the order that they are stated. The printed fields will be identified by their ADABAS field name and column 46-72 data from the ADACMP card images (usually long names). The print length and format will be determined by AUDITRE. Alphanumeric fields will print in character format with a maximum of 85 characters per line. Up to three lines may be used to print large alphanumeric fields. Unpacked and packed (up to eight bytes) will be printed as whole decimal numbers with a possible minus sign. Binary fields will have their last seven bytes printed in hex and in decimal.

Each AUDIT statement's ADABAS-FIELD-LIST is limited to 100 ADABAS fields. With multiple AUDIT statements, this limit is easily avoided.

ALL: This is for use in those cases where listing all elementary fields would be burdensome. In terms of efficiency, ALL is more expensive than selecting a few important fields for auditing. ALL may be imbedded anywhere within a list of field names.

FNR: This states the ADABAS file number. This parameter must be stated, otherwise file zero will be assumed, ending the run as a JCL error for lack of ADACMP card images for file zero.

When more than one file is being audited, two or more reports can be generated, one for each file. If two or more files have related data (Personnel and Finance files), then they can be included in one report, but they must be stated as two or more separate AUDIT statements. When multiple files are referenced in one report, care must be taken to include all appropriate files (INCLUDE statement) for processing.

When a file has several record types (views), each view can be audited, preferably in separate reports referencing the same file, but different fields.

UPDATE= NOTHING This results in a decompression and comparison of Before and After Images for the listed fields. AUDITRE will print the DISPLAY fields, the Images' changed fields, all key fields, and a "RECORD UPDATED" message only when one or more of the listed fields are updated. Stated differently, when none of the listed fields were updated, nothing is printed.

UPDATE = * This results in a "RECORD UPDATED" message and the DISPLAY fields being printed. The Before and After Images are then decompressed and compared for the listed fields. If there are any changes, the changed fields and key fields are printed. If no changes are made to the listed fields, only the key fields are listed.

ADD = LIST This results in a "RECORD ADDED" message and the DISPLAY fields being printed. The After Image is then decompressed for the listed fields. The listed fields are printed if non-null. Key fields are printed.

ADD = * This results in a "RECORD ADDED" message and the DISPLAY fields being printed. The After Image is then decompressed for the listed fields. Only the key fields are listed.

DELETE = LIST This results in a "RECORD DELETED" message and the DISPLAY fields being printed. The Before Image is then decompressed for the listed fields. The listed fields are printed if non-null. Key fields are printed.

DELETE = * This results in a "RECORD DELETED" message and the DISPLAY fields being printed. The Before Image is then decompressed for the listed fields. Only the key fields are listed.

To process the examples shown below, the Compress (ADACMP) file definition card images for file 1 (Personnel File in the examples) must be supplied in the AUDITRE execution JCL. A sample ADACMP for file 1 follows:

FIGURE 8 - SAMPLE ADACMP

```
01,AA,008,B,DE                                  PERSONNEL-NUMBER
01,BA,020,A,NU,DE                               LAST-NAME
01,BB,015,A,NU,DE                               FIRST-NAME
01,BC,001,A,FI                                  INITIAL
01,CA,001,A,NU,DE                               SEX
01,CB,002,U,NU,DE                               AGE
01,CC,010,A,NU,DE                               FAMILY-STATUS
01,CD,002,U,NU,DE                               DEPENDENTS
01,DA,005,U,NU                                  NUMBER
01,DB,020,A,NU,DE                               STREET
01,DC,015,A,NU,DE                               CITY
01,DD,002,A,NU,DE                               STATE
01,DE,005,U,NU,DE                               ZIP-CODE
01,DF,008,A,NU,DE                               PHONE-NUMBER
01,FA,020,A,NU,DE                               JOB
01,FB,006,U,NU,DE                               SALARY
01,FC,006,U,NU                                  COMMISSION
01,GA,002,U,NU                                  YEARS-OF-EDUCATION
01,HA,002,U,NU                                  YEARS-WITH-COMPANY
01,IA,002,U,NU                                  VACATION-DAYS
01,KA,002,U,NU                                  SICK-DAYS
01,LA,030,A,NU,DE                               HOBBY
```

EXAMPLE - ELEMENTARY FIELDS:

```
       AUDIT          BA*,BB,BC,AA*,CA,CB,FB,DA,DB,DC,DD,DE,FNR=1
```

The listed fields from file 1 will be printed in the order AUDITed one line per field, as follows:

```
*   BA=DAVENPORT                                LAST-NAME
*   AA=  HEX 00000000001C4B   DEC 7243          PERSONNEL-NUMBER
 B: FB=046000                                   SALARY
 A: FB=048000                                   SALARY
```

The LAST-NAME (BA) and PERSONNEL-NUMBER (AA) fields are "key fields", denoted on the AUDIT parameter statement and on the print lines with an asterisk (*).

The SALARY field has undergone change (by comparison of Before and After Images), and consequently, the Before (B) and After (A) values are displayed.

EXAMPLE - AUDIT ALL:

```
       AUDIT    ALL,FNR=1
```

This will result in the following field list:

```
  B: FB=046000                                  SALARY
  A: FB=048000                                  SALARY
```

It can be seen that by stating no "key fields", there is no knowledge of which person's salary was updated, other than by ISN.

EXAMPLE - ELEMENTARY FIELDS AND ALL:

```
       AUDIT  BA*,ALL,FNR=1
```

This will result in the following field list:

```
*   BA=DAVENPORT                                    LAST-NAME
 B: FB=046000                                       SALARY
 A: FB=048000                                       SALARY
```

With the key field BA (Last-name), there is identification of the recipient of the salary change.

Assume the FIGURE 8 SAMPLE ADACMP file also contains:

```
01,XY,10,A,MU,NU                                    CHILDREN-NAMES
01,PP,PE                                            SALES-INFO
02,PQ,10,A,NU                                       CUST-ID
02,PR,5,P,NU                                        ITEMS-SOLD
02,PS,7,U,NU                                        DOLLARS-SOLD
02,PT,20,A,NU,MU                                    CUST-PERSONNEL
```

Note that this example now contains an MU field, a PE Group, and fields within the PE, including an MU field within the PE.

EXAMPLE - MU AND PE FIELDS:

MU and PE fields must be specified as occurrence numbers or ranges directly after the field name. For example: XY7-13,PQ12**...**

This may print:

```
B: XY  9=1234
A: XY  9=345
B: XY 12=5212
A: XY 12=80247
```

All changed occurrences within the specified range (XY7-13) will be printed.

EXAMPLE - MU AND PE COUNT FIELDS:

MU and PE count fields must be stated as a field followed by 'C'. A count value will display as 3 bytes when ADABAS 7 PLOG is input and 5 bytes when ADABAS 8 PLOG is input. For example: XYC, XY7-13, PQC...

This may print:

```
B: XYC=9
A: XYC=11
B: XY  9=0
A: XY  9=123
B: XY 11=0
A: XY 11=345
```

This indicates that two additional occurrences of XY have been updated into the record.

When the count field indicates fewer actual occurrences than what are stated for print, the remaining occurrences are decompressed as null values and not printed.

EXAMPLE - MU WITHIN PE FIELDS:

MU within PE fields must be specified by using a # character. For example: PT3-15#4-7 means PT3, MU occurrence number 4, 5, 6, and 7, followed by PT4, MU occurrences number 4, 5, 6, and 7, then PT5, PT6...PT15, MU occurrences number 4, 5, 6, and 7.

This may print:

```
B: PT 14#  6=ABC
A: PT 14#  6=DEFGHI
```

This indicates only one changed occurrence for MU field PT's occurrences 4 through 7 in the periodic group's 3rd through 15th occurrences.

EXAMPLE - MU WITHIN PE COUNT FIELDS:

An MU (within PE) count field must be specified as a field followed by the occurrence number and 'C'. For example: PT7C

This may print:

```
B: PT  7C=1
A: PT  7C=2
```

This indicates a change of an additional occurrence for the MU field PT's 7th occurrence within the PE group.

Each included record's Before and/or After Image will be printed as shown above. If the data is also to be OUTPUT, each field is output according to its FDT (ADACMP) definition length and format. Count fields are output as one binary byte when ADABAS 7 PLOG is input and two binary bytes when ADABAS 8 is input. The exact format of the output record is defined in **Appendix B Output dataset Formats**.

The Personnel file examples used earlier are quite simple. There are no MUs, PEs, or Group Fields in the actual personnel file sample from Software AG. The Finance file (although not a real-life file) presents a much better example of the workings of AUDITRE, especially the AUDIT statement. Following is the FDT (ADACMP) for File 3:

FDT For File 3 (Finance File)

```
01,AA,008,B,DE                          PERSONNEL-NUMBER
01,MC,PE                                MAJOR-CREDIT
  02,CC,018,A,NU,DE                     CREDIT-CARD
  02,CL,004,U,NU,DE                     CREDIT-CARD
  02,CB,004,U,NU,DE                     CURRENT-BALANCE
01,OC,007,A,MU,DE                       OIL-CREDIT
01,NW,008,P,NU,DE                       NET-WORTH
01,CR,002,U,NU,DE                       CREDIT-RATING
01,IP,PE                                INSURANCE-POLICY-TYPES
  02,IC,025,A,MU,NU,DE                  INSURANCE-COMPANY
  02,PA,006,U,MU,NU,DE                  POLICY-AMOUNT
01,CG,016,A,NU,DE                       COLLEGE
01,VC,PE                                VACATION
  02,OV,001,A,NU,DE                     ON-VACATION
01,IV,015,A,NU,DE                       INVESTMENT
01,SV,004,P,NU,DE                       SAVINGS
01,BK,030,A,NU,DE                       BANK
```

Note that this file has PEs, MUs, and MUs within PEs (IC and PA within IP).

Figure 9, which is on the next page, is the result of a SHOW ALL,FNR=3 for only one record on the PLOG, the Before Image for ISN 5.

Note how in Figure 9 all occurrences are processed as 1-10. This is because of the ALL option and the lack of any occurrence information in the FDT. If field MC had been stated as:

```
01,MC,PE(004)
```

only four occurrences of CC would be processed rather than 10, the default. Similarly, the field could be defined as:

```
01,MC,PE(020)
```

to process twenty occurrences.

All actual occurrences up to and including the 20th will be processed. If less than 20 occurrences are present, the remaining are reported as null.

Note that for PE group IP, 10 occurrences of each MU field (IC and PA) within 10 occurrences of IP are reported. This results in 200 lines, many of which are shown as

-
-
-

on Figure 9.

Note that each MU and PE field is preceded by the count field.

FIGURE 9 - SHOW ALL

```
YY-DDD HH-MM-SS   SEQ   HR MI  FNR    ISN    IMAGTYP
11-123 14:19:12      19 14 19     3        5 BEFORE

AA= HEX 000000000186F5  DEC                    100,085      PERSONNEL-NUMBER
MCC=  2                                                     MAJOR-CREDIT
CC  1=DINERS CLUB                                           CREDIT-CARD
CC  2=AMERICAN EXPRESS                                      CREDIT-CARD
CC  3=                                                      CREDIT-CARD
CC  4=                                                      CREDIT-CARD
CC  5=                                                      CREDIT-CARD
CC  6=                                                      CREDIT-CARD
CC  7=                                                      CREDIT-CARD
CC  8=                                                      CREDIT-CARD
CC  9=                                                      CREDIT-CARD
CC 10=                                                      CREDIT-CARD
CL  1=0500                                                  CREDIT-LIMIT
CL  2=0600                                                  CREDIT-LIMIT
CL  3=0000                                                  CREDIT-LIMIT
CL  4=0000                                                  CREDIT-LIMIT
CL  5=0000                                                  CREDIT-LIMIT
CL  6=0000                                                  CREDIT-LIMIT
CL  7=0000                                                  CREDIT-LIMIT
CL  8=0000                                                  CREDIT-LIMIT
CL  9=0000                                                  CREDIT-LIMIT
CL 10=0000                                                  CREDIT-LIMIT
CB  1=0060                                                  CURRENT-BALANCE
CB  2=0025                                                  CURRENT-BALANCE
CB  3=0000                                                  CURRENT-BALANCE
CB  4=0000                                                  CURRENT-BALANCE
CB  5=0000                                                  CURRENT-BALANCE
CB  6=0000                                                  CURRENT-BALANCE
CB  7=0000                                                  CURRENT-BALANCE
CB  8=0000                                                  CURRENT-BALANCE
CB  9=0000                                                  CURRENT-BALANCE
CB 10=0000                                                  CURRENT-BALANCE
OCC=  2                                                     OIL-CREDIT
OC  1=AMOCO                                                 OIL-CREDIT
OC  2=                                                      OIL-CREDIT
OC  3=                                                      OIL-CREDIT
OC  4=                                                      OIL-CREDIT
OC  5=                                                      OIL-CREDIT
OC  6=                                                      OIL-CREDIT
OC  7=                                                      OIL-CREDIT
OC  8=                                                      OIL-CREDIT
OC  9=                                                      OIL-CREDIT
OC 10=                                                      OIL-CREDIT
NW=             3333                                        NET-WORTH
CR=05                                                       CREDIT-RATING
IPC=  1                                                     INSURANCE-POLICY-TYPES
IC  1C=  1                                                  INSURANCE-COMPANY
IC  1#  1=BANKERS LIFE & CASUALTY                           INSURANCE-COMPANY
IC  1#  2=                                                  INSURANCE-COMPANY
IC  1#  3=                                                  INSURANCE-COMPANY
IC  1#  4=                                                  INSURANCE-COMPANY
IC  1#  5=                                                  INSURANCE-COMPANY
IC  1#  6=                                                  INSURANCE-COMPANY
IC  1#  7=                                                  INSURANCE-COMPANY
IC  1#  8=                                                  INSURANCE-COMPANY
IC  1#  9=                                                  INSURANCE-COMPANY
IC  1# 10=                                                  INSURANCE-COMPANY
IC  2C=  0                                                  INSURANCE-COMPANY
IC  2#  1=                                                  INSURANCE-COMPANY
IC  2#  2=                                                  INSURANCE-COMPANY
IC  2#  3=                                                  INSURANCE-COMPANY
•
•
•
```

FIGURE 9 - SHOW ALL (continued)

```
IC  2# 10=                                                  INSURANCE-COMPANY
IC  3C=  0                                                  INSURANCE-COMPANY
IC  3#  1=                                                  INSURANCE-COMPANY
IC  3#  2=                                                  INSURANCE-COMPANY
IC  3#  3=                                                  INSURANCE-COMPANY
IC  3#  4=                                                  INSURANCE-COMPANY
•
•
•
IC  3# 10=                                                  INSURANCE-COMPANY
IC  4C=  0                                                  INSURANCE-COMPANY
•
•
•
IC  5C=  0                                                  INSURANCE-COMPANY
•
•
•
IC 10# 10=                                                  INSURANCE COMPANY
PA  1C=  1                                                  POLICY-AMOUNT
PA  1#  1=035000                                            POLICY-AMOUNT
PA  1#  2=000000                                            POLICY-AMOUNT
•
•
•
PA 10# 10=000000                                            POLICY-AMOUNT
CG=BRIGHAM YOUNG                                            COLLEGE
VCC=  5                                                     VACATION
OV  1=                                                      ON-VACATION
OV  2=                                                      ON-VACATION
OV  3=                                                      ON-VACATION
OV  4=                                                      ON-VACATION
OV  5=Y                                                     ON-VACATION
OV  6=                                                      ON-VACATION
OV  7=                                                      ON-VACATION
OV  8=                                                      ON-VACATION
OV  9=                                                      ON-VACATION
OV 10=                                                      ON-VACATION
IV=CATTLE                                                   INVESTMENT
SV=     150                                                 SAVINGS
BK=MORGAN GUARANTY TRUST, N.Y.                              BANK
```

Figure 9 represents a SHOW of the Before Image for ISN 5, which would normally take about 7 pages to print. Another 7 pages would print the associated After Image. It would be very difficult, time consuming, and error prone to compare Before and After Images in this case. The following AUDIT statement shows the power of AUDITRE:

```
AUDIT                AA*,ALL,FNR=3
```

AUDITRE does the "7 page line-by-line comparison" resulting in the report shown in Figure 10.

FIGURE 10 - AUDIT ALL

```
*     AA=HEX 000000000186F5   DEC 100,085                   PERSONNEL-NUMBER
 B:   NW=3333                                               NET-WORTH
 A:   NW-4444                                               NET-WORTH
```

It is readily apparent that the Net-Worth of person #100085 has been changed.

Figure 11 shows a sample total page that appears for each report:

FIGURE 11 - AUDIT TOTAL PAGE

```
NUMBER OF INPUT RECORDS:    3743

INPUT RECORDS INCLUDED:      175

AUDIT DELETES:                17

AUDIT UPDATES:               122

AUDIT ADDS:                   36
```

Then totals of Updates, Adds, and Deletes of each field in each file listed in the AUDIT statement are shown in Figure 12.

FIGURE 12 - AUDIT File/Field Totals

```
FILE:   3         DELETES:      107        UPDATES:      243        ADDS:       88

FIELD   LONG NAME             OCC FROM  OCC TO  OCC FROM  OCC TO   DELETES  UPDATES  ADDS
AA      PERSONNEL-NUMBER                                           107      0        88
MCC     MAJOR-CREDIT                                               107      15       88
CC      CREDIT-CARD           1         4                          107      15       88
CL      CREDIT-LIMIT          1         4                          107      15       88
CB      CURRENT-BALANCE       1         4                          107      15       88
OCC     OIL-CREDIT                                                 107      3        88
OC      OIL-CREDIT            1         10                         107      3        88
NW      NET-WORTH                                                  107      37       88
CR      CREDIT-RATING                                              107      143      88
IPC  INSURANCE-POLICY-TYPES                                        107      2        88
ICC     INSURANCE-COMPANY     1         1                          107      5        88
IC   INSURANCE-COMPANY       1         1       1         10       107      5        88
PAC     POLICY-AMOUNT         1         1                          107      7        88
PA      POLICY-AMOUNT         1         1       1         10       107      7        88
CG      COLLEGE                                                    107      38       88
VCC     VACATION                                                   107      26       88
OV      ON-VACATION                                                107      24       88
IV      INVESTMENT                                                 107      77       88
SV      SAVINGS                                                    107      22       88
BK      BANK                                                       107      13       88


FILE:   1         DELETES:       36        UPDATES:      127        ADDS:       41

FIELD   LONG NAME             OCC FROM  OCC TO   OCC FROM OCC TO   DELETES  UPDATES  ADDS
AA    PERSONNEL-NUMBER                                             3        0        41
BA    LAST-NAME                                                    36       0        41
BB    FIRST-NAME                                                   36       0        41
BC    INITIAL                                                      36       0        41
CB    AGE                                                          36       106      41
LA    HOBBY                                                        36       91       41
```

III.2.9 CONTROL

The CONTROL statement defines the fields that control the sequence and control-break(s) for the summary report.

SYNTAX

```
CONTROL        FIELD[,FIELD]...
```

The fields specify the major-to-minor sequence and control break fields for the report. Up to 3 control fields may be stated.

EXAMPLES

```
CONTROL        FNR
```

The summary report from the Protection Log will be in ascending order of FNR. Summarizations will be shown for each FNR value, and a grand total will be shown.

```
CONTROL        RUI,FNR,HOUR
```

The summary report from the Protection Log will be in ascending order of HOUR within FNR within RESTART-USERID. Summarizations will be shown for each HOUR within each FNR within each RESTART-USERID, and a grand total will be shown.

```
CONTROL        DEPT,FNR
```

The summary report will be in ascending order of FNR within DEPT (a derived field). Summarization will be shown for each FNR used by each department, and a grand total will be shown.

III.2.10 OUTPUT

The OUTPUT statement specifies that AUDITRE should output certain data onto a sequential dataset.

Certain Protection Log input records can be selected (INCLUDE/EXCLUDE) for detail reports and for outputting to a sequential dataset.

There are three types of AUDITRE output:

- Compressed detail data
- Decompressed detail data in flat-file format
- Summary data in flat-file format

Compressed Detail Data

The OUTPUT statement COMPRESS=YES option is used to specify that the output detail data should remain in compressed format in the same format as the Protection Log. Therefore, compressed output from AUDITRE may subsequently be used as input to later AUDITRE executions with the specification of INPUT LOGTYPE=PROTECTION.

Decompressed Detail Data in Flat-file Format

The OUTPUT statement COMPRESS=NO option is used to specify that the output detail data should be decompressed. Output data that has been decompressed does not resemble the original Protection Log input, and therefore, cannot be reinput to AUDITRE as INPUT LOGTYPE=PROTECTION. Refer to **Appendix B Output Dataset Formats** for this decompressed data output format.

Output files for SHOW reports and AUDIT reports have identical formats (i.e., the listed fields are decompressed in exactly the same manner). However, AUDITed OUTPUTs may contain fewer records because only records that have Updates to the listed image fields are actually output. Adds are always output, considering that listed fields changed from nothing to something. Similarly, Deletes are always output, considering that listed fields changed from something to nothing.

Summary Data in Flat-file Format

If the OUTPUT statement is specified for a summary report, the count field shown for each lowest control field on the print line on the summary report will be placed in a record on a sequential dataset. Refer to **Appendix B Output Dataset Formats** for this summary data output format.

"Flat-file" output sequential datasets can be further processed using user-written COBOL, NATURAL, etc. programs, or the data can be placed into an ADABAS file by the Mass Update Utility (ADALOD) or a user program if desired.

SYNTAX

```
        OUTPUT      LIMIT      =  {  99999999
                                     numeric-value     }

                    ID         =  {  character-string
                                     USER              }

                    COMPRESS   =  {  NO
                                     YES    }

                    YEARFMT    =  {  2
                                     4      }

                    SYSNO      =  {  011                 (VSE only)
                                     013
                                     021
                                     023
                                     031
                                     033
                                     041
                                     043
                                     051
                                     053    }

                    UNLOAD     =  {  YES                 (VSE and
                                     NO     }             VM only)
```

LIMIT: This specifies a limit to the number of output records.

ID: This identifies up to 8 characters that can be used to identify the summary output file records. This is especially important for VSE as all summary records are output onto one sequential dataset.

COMPRESS: A "YES" specification for a Detail Report will cause output records in the same Protection Log compressed form as the input.

A "NO" specification for a Detail Report will cause output records in decompressed flat file form.

Refer to **Appendix B Output Dataset Formats**.

YEARFMT: The year format in a PLOG Output Dataset of decompressed form (COMPRESS=NO) can be two digits (YEARFMT=2) or four digits (YEARFMT=4).

The year format in a CLOG Output Dataset for Summary Reports can be two digits (YEARFMT=2) or four digits (YEARFMT=4).

Refer to **Appendix B Output Dataset Formats**.

SYSNO: For VSE , SYSNO=011, 021, 031, 041, and 051 refer to Tape output; and SYSNO=013, 023, 033, 043, and 053 refer to Disk output. For OS and VM, this is accomplished through JCL. Refer to **Section IV.5.2, Execution Procedure for VSE**.

UNLOAD: For VSE and VM, this specifies whether or not to unload the output tape. The tape should be rewound but not unloaded when the AUDITRE run is followed by another job that is to process the AUDITRE output tape.

EXAMPLES

```
INPUT          LOGTYPE=PROTECTION
REPORT         TYPE=DETAIL
INCLUDE        FNR=1
DISPLAY        ...
SHOW           AA,BB,CC,DD,FNR=1
OUTPUT
```

In this example, all included Protection Log detail input records will be output in a different (decompressed) form to a sequential dataset.

```
INPUT          LOGYTPE=PROTECTION
REPORT         TYPE=DETAIL
INCLUDE        FNR=(30,31,32)
DISPLAY        ...
SHOW           ...
OUTPUT         COMPRESS=YES
```

In this example, all included Protection Log detail input records will be output to a sequential dataset in the same format as the input. This is a method of reducing PLOG data to only those files or users to be AUDITed at a later date.

```
INPUT          LOGTYPE=PROTECTION
REPORT         TYPE=SUMMARY
CONTROL        FNR,RUI
OUTPUT         ID=FNRRUI
```

In the example, a summary report is produced. For each report line showing data for the lowest control field (RUI), a record will be output to a sequential dataset. Each record on the output dataset will contain "FNRRUI" in the "ID" field.

Refer to **Appendix B Output Dataset Formats** for exact format of these outputs.

III.3 Parameter Statement Order

Parameter statements for log processing should be stated in the following order:

First: INPUT

Then: Derived Fields. For each derived field:
- FIELD Statement
- VALUE Statement(s)

Then: As many reports as desired

For each DETAIL report:
- REPORT statement
- INCLUDE/EXCLUDE statement(s), if desired
- DISPLAY statement(s), if desired
- SHOW or AUDIT statements, if desired
- OUTPUT statement, if desired

For each SUMMARY report:
- REPORT statement
- INCLUDE/EXCLUDE statement(s), if desired
- CONTROL statement, if desired
- OUTPUT statement, if desired

For each statement having multiple fields, the fields will be shown on the report in the order stated in the parameter statements.

III.4 AUDITRE Reports and Outputs

III.4.1 Report Headings

Parameters on the Report statement indicate the one or two heading lines of up to 60 characters each, which AUDITRE will place at the top of every report page. Also, the AUDITRE version number, the day of the AUDITRE run (SUN, MON, etc.), the date (YY-MM-DD), the time (HH:MM:SS), and the page number will be displayed at the top of each page.

III.4.2 Report Dimensions

The LINE-SIZE setting of 133 can only be changed to a higher number. For VSE, only 121 characters per line will be printed. The PAGE-SIZE setting of 55 can be overridden to form reports more suitable for screen viewing (e.g., 24 lines).

III.4.3 Report Destinations

AUDITRE reports are not output directly to a printer, but to datasets that may or may not be printed or viewed. In OS or VM, reports are output to datasets AUDPRT00, AUDPRT01, etc. up to AUDPRT99. In VSE, reports are output to SYS005, 015, 025, 035, and 045. Note that AUDPRT99 is reserved for use to direct the end of detail report statistics and end of summary report statistics, as well as the end of job statistics in AUDPRT99 (requires optional run time parameter PARM=SUMMARY).

AUDPRT00 will always be a report showing the parameters input to AUDITRE and a few useful statistics, such as the number of records input, the number of reports, and the execution time (wall-clock time). If the AUDITRE parameters are in error, the report will show the parameters and any error messages. Refer to **Section V.1 Error Messages**.

AUDPRT01 will receive the first DETAIL Report. Each subsequent DETAIL Report will cause the dataset number to increase by one.

SUMMARY Reports are output to the same dataset as the preceding DETAIL Report.

All reports are output in Fixed Length record format with the first character of each record being the standard ASCII printer carriage control character for top-of-page, single, double, and triple spacing. No "overprinting" is used in AUDITRE.

III.4.4 Column Headings, Spacing, Skipping

AUDITRE places headings on top of each page above the fixed or derived field that is being printed. The headings are descriptive without being excessive.

All report heading lines are printed at the top of the page. Field heading lines are double spaced. Report detail lines are single spaced. Summary report lines are single spaced except for control break lines (double spaced) and the grand total line (triple spaced).

One space will separate each displayed field or its header (whichever is larger) on the report.

III.4.5 Detail Reports - Contents

The log records printed on a Detail Report are determined by the INCLUDE and EXCLUDE parameters, if stated. If none are stated, all records will be printed. Records are always printed in the order in which they are input.

The fixed or derived fields that will be printed are determined by the DISPLAY statement field list. The fields are displayed in the order stated in the DISPLAY statement. No totals or summarization of any fields are shown.

Decompressed ADABAS fields are displayed usually one per line in the order stated on the SHOW or AUDIT statement. Binary fields are printed in hex and decimal. Long fields are printed on multiple lines.

SHOW and AUDIT fields are identified by the ADABAS-FIELD-NAME (2-character name), occurrence number(s) if appropriate, and a long name (27 characters) from the ADACMP card images.

AUDIT fields are identified as "B:" for the Before value and "A:" for the After value. Key fields are identified by an "*".

After all detail data is displayed, the report displays the total number of input records, records included, and records output to a sequential dataset, if any.

AUDIT reports conclude with a total of Updates, Adds, and Deletes for each field stated on each AUDIT statement for each file. If AUDITRE is run with the optional run time parameter PARM=SUMMARY and AUDPRT99 is specified in the JCL, the end of detail report statistics will be written to AUDPRT99, as well as at the end of the detail report.

III.4.6 Summary Report - Contents

The input records processed can be limited by the INCLUDE and EXCLUDE parameters, if stated. If none are stated, all records are processed.

Summary Reports are not printed until all input records are processed.

The second Summary Report heading line contains the earliest and latest log dates and times.

The order of the report lines is determined by the CONTROL parameter field(s) in ascending order of Field-1, and then ascending order of Field-2 within Field-1, etc., up to three levels (fields).

Control totals are printed at each break in value of each CONTROL parameter field. A grand total line is printed after the last control break line.

The report ends with a display of the total number of input records, records included, and records output to a sequential dataset, if any. If AUDITRE is run with the optional run time parameter PARM=SUMMARY and AUDPRT99 is specified in the JCL, the end of summary report statistics will be written to AUDPRT99, as well as at the end of the summary report.

Refer to **Appendix B Output Dataset Formats** for a definition of the output format for Summary Reports for which an OUTPUT statement is included.

SECTION IV

INSTALLATION AND OPERATIONS

IV.1 Introduction

AUDITRE may be installed and executed in the following environments: Z/OS, VM, or VSE (VSE/ESA or Z/VSE) on IBM 390 and compatible mainframes. AUDITRE is also installable on certain BS2000/OSD systems.

This section describes the installation process for AUDITRE. It is divided into subsections, one for each operating system, as follows:

Z/OS Installation (refer to **Section IV.2 Z/OS Installation**)

VSE (Z/VSE) Installation (refer to **Section IV.3 VSE (Z/VSE) Installation**)

VM Installation (refer to **Section IV.4 VM Installation**)

BS2000/OSD Installation (refer to **Section IV.5 BS2000 Installation**)

AUDITRE requires no zaps to any operating system, teleprocessing system, or to ADABAS, NATURAL, or their associated software.

AUDITRE batch facilities are user-parameter driven with no extensive user-coding required.

This version of AUDITRE can be used with ADABAS 7 or 8 Protection Logs. There can be no "mixing" of modules from various AUDITRE versions, or the results will be unpredictable. If the PLOG input includes spanned block records, the ADABAS parameters creating the PLOG must be defined with SRLOG=ALL to ensure the entire Before and/or After Image is included on the PLOG. If the entire image is not included on the PLOG, the results will be unpredictable.

IV.2 Z/OS Installation

AUDITRE is distributed via a zip file or on a 3490 cartridge.

The AUDITRE JCL dataset contains sample JCL to assist in installation and operation of AUDITRE. The following z/OS specific members are contained in this file:

```
JOSUPDTE       (JCL for Building the Load PDS)
JOSPLOG        (Sample JCL to run AUDITRE for Protection Log Analysis)
PPLOG          (Parameters for Protection Log Analysis)
```

The AUDITRE LOAD dataset contains the following load modules:

- AUDITRE7 - batch multi-purpose program for ADABAS version 7.4.x PLOGs.
  The AUDOUT* data produced will have a one-byte field for MU and PE occurrence counts, MU and PE count fields will occupy three bytes in Auditre reports.
- AUDITRE8 – batch multi-purpose program for ADABAS version 8.x PLOGs. The AUDOUT* data produced will have a two-byte field for MU and PE occurrence counts, MU and PE counts will occupy five bytes in Auditre reports.
- AUDITRE - new module as of Auditre v412.
  Batch multi-purpose program for PLOGs from ADABAS 7.4.x and 8.x
  Note: This module should NOT BE defined on the EXEC statement in JCL It will be dynamically loaded by the front-end module AUDITRE7 or AUDITRE8.

The following steps are required for the Z/OS installation of AUDITRE. It is important to use this list as a guide to be sure that the installation is complete. Each step is covered in detail in this section.

1. Load the distribution to the z/OS system.
2. Apply AUDITRE ZAPs

IV.2.1 Loading files from zip distribution

FTP must be running on the mainframe in order to transfer these files.

Contents of Auditre Zip file:

File	Description
AUDvrs.PDF	Auditre Manual
AUDvrsRN.PDF	Auditre Release Notes
AUD.Vvrs.JCL.XMIT	Auditre Execution JCL
AUD.Vvrs.LOD.XMIT	Auditre Load Library
AUD.Vvrs.README.TXT	Auditre Install instructions/fix descriptions

IV.2.1.1 Allocate datasets

To load these datasets to the mainframe, allocate the following sequential datasets:

Dataset	DCB Information
TEMP.AUD.Vvrs.JCL	RECFM=FB,LRECL=80,BLKSIZE=3120, SPACE=(CYL,(1,1))
TEMP.AUD.Vvrs.LOD	RECFM=FB,LRECL=80,BLKSIZE=3120, SPACE=(CYL,(6,3))
PREFIX.AUD.Vvrs.README	RECFM=FB,LRECL=80,BLKSIZE=3120,

SPACE=(TRK,(2,2))

IV.2.1.2 Transfer datasets to mainframe

In binary mode:
Transfer the AUD.Vvrs.JCL.XMIT file to the TEMP.AUD.Vvrs.JCL dataset.
Transfer the AUD.Vvrs.LOD.XMIT file to the TEMP.AUD.Vvrs.LOD dataset.

IV.2.2.3 Receive datasets

Once the binary transfers are complete, do one of the following 2 options:

1. select option 6 from the ISPF Primary Option window. From the TSO Command line, enter the following command:

```
receive indataset('TEMP.AUD.Vvrs.JCL')
DSNAME(PREFIX.AUD.Vvrs.JCL)
receive indataset('TEMP.AUD.Vvrs.LOD')
DSNAME(PREFIX.AUD.Vvrs.LOAD)
```

   When prompted for the restore parameters or 'DELETE' or 'END', press enter.

2. Execute the following sample JCL to RESTORE the dataset:

```
JOBCARD
//RECEIVE  EXEC PGM=IKJEFT01,DYNAMNBR=75,TIME=1440,REGION=6M
//INDD2    DD DISP=(SHR,DELETE,DELETE),DSN=TEMP.AUD.Vvrs.JCL
//INDD3    DD DISP=(SHR,DELETE,DELETE),DSN=TEMP.AUD.Vvrs.LOD
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTERM  DD DUMMY
//SYSUADS  DD DSN=SYS1.UADS,DISP=SHR
//SYSLBC   DD DSN=SYS1.BRODCAST,DISP=SHR
//SYSTSIN  DD *
RECEIVE INDDNAME(INDD1)
DSNAME(PREFIX.AUD.Vvrs.JCL)
RECEIVE INDDNAME(INDD2)
DSNAME(PREFIX.AUD.Vvrs.LOAD)
/*
```

IV.2.2 Loading files from Tape distribution

The AUDITRE tape volume serial number is VOLSER = AUDvrs, unless otherwise noted on the tape sticker or in the release notes. The tape contains two files.

The JCL dataset requires one track of disk space.

The LOAD dataset requires one cylinder of disk space.

IV.2.2.1 Allocate Space

Sample OS JCL to allocate and catalog the AUDITRE source, object, and load libraries follows:

```
//ALLOC1      EXEC   PGM=IEFBR14
//AUDITRE1    DD     DSN=AUDITRE.Vvrs.SOURCE,
//                   SPACE=(3120,(5,1,1)),
//                   UNIT=SYSDA,VOL=SER=AUDvrs,
//                   DISP=(,CATLG,DELETE),
//                   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120,DSORG=PO)
//*
//ALLOC2      EXEC   PGM=IEFBR14
//AUDITRE2    DD     DSN=AUDITRE.Vvrs.LOAD,
//                   SPACE=(CYL,(3,2,10)),
//                   UNIT=SYSDA,VOL=SER=AUDvrs,
//                   DISP=(,CATLG,DELETE),
//                   DCB=(RECFM=U,LRECL=0,BLKSIZE=27998,DSORG=PO)
//*
```

IV.2.2.2 Load Datasets to Disk

Source Modules

Using the IEBUPDTE utility, file one of the installation tape must be loaded to disk. Sample JCL to load this file follows:

```
//LOADSRC     EXEC   PGM=IEBUPDTE
//SYSPRINT    DD     SYSOUT=A
//SYSUT1      DD     DSN=AUDITRE.Vvrs.SOURCE,DISP=OLD
//SYSUT2      DD     DSN=AUDITRE.Vvrs.SOURCE,DISP=OLD
//SYSIN       DD     DSN=AUDITRE.Vvrs.SOURCE,DISP=(OLD,PASS),UNIT=3490,
//                   LABEL=(1,SL),VOL=SER=AUDvrs
```

Load Modules

Using the IEBCOPY utility, file two of the installation tape must be loaded to disk. Sample JCL to load this file follows:

```
//LOADLOD     EXEC   PGM=IEBCOPY
//SYSUT3      DD      UNIT=SYSDA,SPACE=(CYL,(3,1))
//SYSUT4      DD      UNIT=SYSDA,SPACE=(CYL,(3,1))
//OUTDSN      DD     DSN=AUDITRE.Vvrs.LOAD(OLD,KEEP)
//INDSN       DD     DSN=AUDITRE.Vvrs.LOAD,DISP=(OLD,PASS),UNIT=3490,
//                   LABEL=(2,SL),VOL=SER=AUDvrs
//SYSPRINT    DD     SYSOUT=*
//SYSIN       DD     *
 COPY INDD=INDSN,OUTDD=OUTDSN//
```

IV.2.3 Apply AUDITRE ZAPs

Before proceeding with the AUDITRE installation, apply a ZAP to set the expiration date. AUDITRE will arrive expired and will need a ZAP to function. There may be some corrections that must be applied to the distributed code. These corrections are supplied as official fixes, which are shipped with the AUDITRE release. Refer to the README dataset for details on current official fixes.

IV.3 VSE (Z/VSE) Installation

IV.3.1 Loading Files from ZIP Distribution

FTP must be running on the mainframe in order to transfer these files.
Contents of Auditre Zip file:

File	Description
AUDvrs.PDF	Auditre Manual
AUDvrsRN.PDF	Auditre Release Notes
AUD.Vvrs.LIBR	Auditre object, JCL and source modules

IV.3.2 Transfer Datasets to Mainframe

Transfer the AUD.Vvrs.LIBR file to the TEMP.AUD.Vvrs.LIBR dataset in binary mode.

Code and submit a LIBR job using the FTP'd LIBR dataset as SYSIPT to CATALOG the sample JCL, objects and sample source members to a private library.

```
* $$ JOB JNM=AUDLOAD,CLASS=0,DISP=D
* $$ LST DISP=D,CLASS=A
*
* ********************************************************
* VSE JCL TO INSTALL (LOAD) AUDITRE JCL &TEXT
* INTO A VSE LIBRARY USING LIBR
* USE FTP FILE FOR INPUT TO THIS JOB
* ********************************************************
*
// JOB AUDLOAD RESTORE AUDITRE VSE SUBLIB
// ASSGN SYSIPT,cuu POINT TO FTP'E LIBR File
// DLBL IJSYSIN,'TEMP.AUD.Vvrs.LIBR'
/// DLBL AUDLIB,'TSI.AUDITRE.LIBRARY',,SD <- OMIT IF STANDARD LABELS
// EXTENT ,vol,1,0,nnnn,ttt               <- OMIT IF STANDARD LABELS
// EXEC LIBR,PARM='ACCESS S=AUDLIB.Vvrs'
/*
/&
* $$ EOJ
```

IV.3.2 Installation Tape

AUDITRE for VSE (Z/VSE) environments is described in this section.

AUDITRE is distributed on a 3490 cartridge. The AUDITRE tape volume serial number is VOLSER = AUDvrs unless otherwise noted on the tape sticker or in the release notes.

The tape contains one file. The file is a source and object file that contains several members to assist in installation and operation of AUDITRE in the VSE (VSE/ESA) environment. This file is blocked at 6480 bytes. The following source members are contained in this file:

```
    JVSLINK7       (JCL FOR AUDITRE LINK-edit to be used with ADABAS V7.4.x PLOGS)
   JVSLINK8        (JCL FOR AUDITRE LINK-edit to be used with ADABAS V8.x PLOGS)
  VSLINKA          (JCL FOR AUDITRE LINK-edit for the primary AUDITRE phase loaded
                        by either AUDITRE7 or AUDITRE8)
  JVSMSHP          (Sample JCL to define Auditre to a MSHP history file)
```

```
JVSPLOGD       (JCL for Protection Log Analysis from disk PLOG input)
JVSPLOGT       (JCL for Protection Log Analysis from tape PLOG input)
PPLOGVD        (Parameters for Protection Log Analysis from disk PLOG input)
PPLOGVT        (Parameters for Protection Log Analysis from tape PLOG input)
```

Approximately 16 object members are contained in this file. The Link-edit step, which is presented later, combines specific object members into the execution module necessary in an AUDITRE environment, including:

After execution of the three link-edit jobs the VSE sub-library contains the following PHASE members:

- AUDITRE7 - batch multi-purpose program for ADABAS version 7.4.x PLOGs. The AUDOUT* data produced will have a one-byte field for MU and PE occurrence counts, MU and PE count fields will occupy three bytes in Auditre reports.
- AUDITRE8 – batch multi-purpose program for ADABAS version 8.x PLOGs. The AUDOUT* data produced will have a two-byte field for MU and PE occurrence counts, MU and PE counts will occupy five bytes in Auditre reports.
- AUDITRE -Batch multi-purpose program for PLOGs from ADABAS 7.4.x and 8.x
  Note: This module should NOT BE defined on the EXEC statement in JCL It will be dynamically loaded by the front-end module AUDITRE7 or AUDITRE8.

IV.3.2 Installation Steps

The following steps are required for the VSE (ESA) installation of AUDITRE. It is important to use this list as a guide to be sure that the installation is complete. Each step is covered in detail in this section.

1. Define the AUDITRE Vv.r.s library and sub-library.
2. Copy the tape to disk.
3. Define Auditre to the VSE MSHP history file using the JCL in JVSMSHP.J as a guide.
4. Apply the license or trial zap to AUDBAUTH.OBJ as supplied and directed by TSI.
5. Link-edit the AUDITRE Components.
6. Apply AUDITRE ZAPs if any and re-link the Auditre PHASEs.

IV.3.2.1 Define the AUDITRE Vv.r.s Library and Sub-library

Use a library that has been included in VSE *Standard Labels.* The example uses ‘TSI.AUD.LIBRARIES’ for the library data set name. The sub-library name is ‘AUDLIB.Vvrs’.

IV.3.2.2 Copy the Tape to Disk

The following example JCL can be used once the appropriate modifications to job name, volume serial numbers, extents, tape cuu, library names, etc. have been made.

```
* $$ JOB JNM=AUDLOADT,CLASS=4,DISP=D,PRI=3
* $$ LST LST=SYSLST,CLASS=Q
// JOB AUDLOADT
/*   THIS JOB COPIES THE AUD SUBLIBRARY AUDLIB.Vvrs TO VSE.
/*   YOU MUST HAVE ALREADY CREATED THE AUD SUBLIBRARY FOR THIS
/*   VERSION OF AUD.  IT IS RECOMMENDED THAT THE AUD LIBRARY BE
/*   ADDED TO THE STANDARD LABELS, IN WHICH CASE THE DLBL AND
/*   EXTENT FOR IT SHOULD BE OMITTED.
// DLBL AUDLIB,'TSI.AUD.LIBRARIES',,SD    <--OMIT IF IN STANDARD LABELS
// EXTENT ,SYSWK1,1,0,18675,4500          <--OMIT IF IN STANDARD LABELS
/* EXEC LIBR
/* ACCESS  SUBLIB=AUDLIB.Vvrs
/* DELETE  *.*
/*
// ASSGN SYSIPT,631
// MTC REW,SYSIPT
// MTC FSF,SYSIPT,1
// EXEC LIBR,PARM='ACCESS S=AUDLIB.Vvrs'
// MTC REW,SYSIPT
/*   ------------------------------------------------------
/*
/&
* $$ EOJ
```

IV.3.2.3 Define Auditre to the MSHP history file

Zaps for problem correction and for the license of the Auditre product will be provided as corrections to Auditre VSE object modules. This requires Auditre to be defined to the MSHP history file. The following JCL may be used after changes for site-specific requirements. Member JVSMSHP.J is provided in the Auditre distribution as a template.

```
* $$ JOB JNM=JVSMSHP,CLASS=0,DISP=D,PRI=3
* $$ LST CLASS=Q
// JOB JVSMSHP
/*   THIS JOB DEFINES AUDITRE V521 TO MSHP
/* ------------------------------------------------------
// OPTION LOG
// EXEC MSHP
 ARCHIVE AUD521
 COMPRISES 9999-AUD-00
 RESOLVES 'TREEHOUSE SOFTWARE - AUDITRE 5.2.1'
 ARCHIVE 9999-AUD-00-521
 RESIDENCE PRODUCT=AUD521             -
            PRODUCTION=AUDLIB.V521 -
            GENERATION=AUDLIB.V521
/*
/&
* $$ EOJ
```

Note the above JCL requires that the MSHP history file and the Auditre library be added to standard labels. If they are not, appropriate ASSGN, DLBL and EXTENT statements must be added to the JCL member before it is submitted.

IV.3.2.4 Apply the License Zap Supplied by TSI

Treehouse Software will provide an appropriate MSHP zap to allow proper execution of Auditre at your site. Refer to the provided zap for further details.

IV.3.2.5 Link Edit the AUDITRE Components

Make the site-dependent changes to the link-edit JCL from the AUDITRE sub-library AUDLIB.Vvrs, members *JVSLINKA.,J JVSLINK7.J* and *JVSLINK8.J.*

Run the link edit job. Verify the highest completion code to be 2.

Submit JVSLINK7 to place the AUDITRE7 phase in AUDLIB.Vvrs.
Submit JVSLINK8 to place the AUDITRE8 phase in AUDLIB.Vvrs.
Submit JVSLINKA to place the AUDITRE phase in AUDLIB.Vvrs.

Following is an example of JVSLINKA, JVSLINK7 and JVSLINK8:

Primary AUDITRE PHASE

```
* $$ JOB JNM=JVSLINKA,CLASS=0,DISP=D
* $$ LST DISP=D,CLASS=A
// JOB JVSLINKA        LINKEDT AUDITRE VSE SUBLIB
/*
/*  LINKEDT AUDITRE COMPONENTS CREATING AUDITRE, THE MAIN
/*  PHASE EXECUTED TO RUN AUDITRE
/*
/* ********************************************************************
/* VSE JCL TO LINKEDIT AUDITRE
/* ********************************************************************
/*  IT IS RECOMMENDED THAT THE AUDITRE LIBRARY BE ADDED TO THE
/*  STANDARD LABELS, IN WHICH CASE THE CLBL AND EXTENT CAN BE
/*  OMITTED.
/*
// DLBL AUDLIB,'TSI.AUDITRE.LIBRARY',,SD <- OMIT IF IN STANDARD LABELS
// EXTENT ,SYSWK1,1,0,18675,4500         <- OMIT IF IN STANDARD LABELS
// LIBDEF OBJ,SEARCH=AUDLIB.Vvrs
// LIBDEF PHASE,CATALOG=AUDLIB.Vvrs
// OPTION LIST,SXREF,CATAL
   ACTION  MAP,CANCEL,ERRLMT(5)
   PHASE AUDITRE,*
  MODE    AMODE(24),RMODE(24)
  INCLUDE AUDBDRVS
  INCLUDE AUDXPLOG
  INCLUDE TSIXPLOG
  INCLUDE AUDBBEGN
  INCLUDE AUDBAUTH
  INCLUDE AUDBCONT
  INCLUDE AUDBDETL
  INCLUDE AUDBDOPL
  INCLUDE AUDBMISC
  INCLUDE AUDBPAP1
  INCLUDE AUDBPAP2
  INCLUDE AUDBSTPR
  INCLUDE AUDBSUMP
  INCLUDE AUDBEND
  ENTRY   AUDITRE
// EXEC LNKEDT,SIZE=256K
/*
/&
* $$ EOJ
```

ADABAS v7.4.x

```
* $$ JOB JNM=JVSLINK7,CLASS=0,DISP=D
* $$ LST DISP=D,CLASS=A
// JOB JVSLINK7        LINKEDT AUDITRE VSE SUBLIB
/*
/*  LINKEDT AUDITRE COMPONENTS CREATING AUDITRE7, TO BE USED WITH
/*  PLOGS CREATED IN ADABAS VERSIONS V7.4.X.
/*
/* ****************************************************************
/* VSE JCL TO LINKEDIT AUDITRE
/* ****************************************************************
/*  IT IS RECOMMENDED THAT THE AUDITRE LIBRARY BE ADDED TO THE
/*  STANDARD LABELS, IN WHICH CASE THE CLBL AND EXTENT CAN BE
/*  OMITTED.
/*
// DLBL AUDLIB,'TSI.AUDITRE.LIBRARY',,SD <- OMIT IF IN STANDARD LABELS
// EXTENT ,SYSWK1,1,0,18675,4500         <- OMIT IF IN STANDARD LABELS
// LIBDEF OBJ,SEARCH=AUDLIB.Vvrs
// LIBDEF PHASE,CATALOG=AUDLIB.Vvrs
// OPTION LIST,SXREF,CATAL
   ACTION  MAP,CANCEL,ERRLMT(5)
   PHASE AUDITRE7,*
   MODE    AMODE(24),RMODE(24)
   INCLUDE AUDITRE7
   ENTRY AUDITRE7
// EXEC LINKEDT,SIZE=100K
/*
/&
* $$ EOJ
```

ADABAS v8.x

```
* $$ JOB JNM=JVSLINK8,CLASS=0,DISP=D
* $$ LST DISP=D,CLASS=A
// JOB JVSLINK8        LINKEDT AUDITRE VSE SUBLIB
/*
/*  LINKEDT AUDITRE COMPONENTS CREATING AUDITRE8, TO BE USED WITH
/*  PLOGS CREATED IN ADABAS VERSIONS V8.X.
/*
/* ****************************************************************
/* VSE JCL TO LINKEDIT AUDITRE
/* ****************************************************************
/*  IT IS RECOMMENDED THAT THE AUDITRE LIBRARY BE ADDED TO THE
/*  STANDARD LABELS, IN WHICH CASE THE CLBL AND EXTENT CAN BE
/*  OMITTED.
/*
// DLBL AUDLIB,'TSI.AUDITRE.LIBRARY',,SD <- OMIT IF IN STANDARD LABELS
// EXTENT ,SYSWK1,1,0,18675,4500         <- OMIT IF IN STANDARD LABELS
// LIBDEF OBJ,SEARCH=AUDLIB.Vvrs
// LIBDEF PHASE,CATALOG=AUDLIB.Vvrs
// OPTION LIST,SXREF,CATAL
   ACTION  MAP,CANCEL,ERRLMT(5)
   PHASE AUDITRE8,*
   MODE    AMODE(24),RMODE(24)
   INCLUDE AUDITRE8
   ENTRY AUDITRE8
// EXEC LINKEDT,SIZE=100K
/*
/&
* $$ EOJ
```

IV.3.2.4 Apply AUDITRE ZAPs

Before proceeding with the AUDITRE installation, apply a ZAP to set the expiration date. AUDITRE will arrive expired and will need a ZAP to function. There may be some corrections that must be applied to the distributed code. These corrections are supplied as official fixes, which are shipped with the AUDITRE release. Refer to the README dataset for details on current official fixes. The README file can be found in AUDLIB.Vvrs sub-library member README.VSE.

IV.4 VM Installation

IV.4.1 Installation Tape

AUDITRE is distributed on a 3490 cartridge. The tape contains one dataset. The dataset contains both source and object (TEXT) files in VM TAPE DUMP format. The following source files are included:

```
JVMGEN7      EXEC        (EXEC for creation of AUDITRE7 module)
JVMGEN8      EXEC        (EXEC for creation of AUDITRE8 module)
JVMPLOG      EXEC        (EXEC for Protection Log Analysis)
PPLOG        AUDPARM     (Parameters for Protection Log Analysis)
AUDITRE7     TXTLIB      (Object code for AUDITRE7)
AUDITRE8     TXTLIB      (Object code for AUDITRE8)
```

The installation steps, which are presented later, combine specific object members into the appropriate execution module necessary in an AUDITRE environment, including:

AUDITRE7 - batch multi-purpose program for ADABAS version 7.4.x PLOGs
AUDITRE8 - batch multi-purpose program for ADABAS version 8.x PLOGs

IV.4.2 Installation Steps

The following steps are required for the VM installation of AUDITRE. It is important to use this list as a guide to be sure that the installation is complete. Each step is covered in detail in this section.

1. Assign and allocate required mini-disk space.
2. Load the first file from the release tape to the assigned mini-disk.
3. Build the AUDITRE module.
4. Apply AUDITRE ZAPs.

IV.4.2.1 Allocate VM/CMS Mini-disk Space

Installation of AUDITRE will require the equivalent of a one-cylinder 3380 mini-disk formatted in 1024 byte blocks.

For FBA allocation, use 8000 (512K) blocks.

Sample CMS Command to format a mini-disk:

```
FORMAT   301    f
```
(where "301" is the address and "f" is the Disk File Mode)

IV.4.2.2 Load to Allocated Mini-disk

Sample CMS Command to Load the AUDITRE files to disk:

```
TAPE LOAD  *  *  f
```
(where "f" is the Disk File Mode)

IV.4.2.3 Build AUDITRE Modules

The AUDITRE modules are generated by running the JVMGEN EXEC, which has been loaded from the first file of the release tape.

Before executing JVMGEN, be prepared to answer a prompt for the file mode desired for the AUDITRE MODULE file.

Simply enter the command “JVMGEN”. The EXEC will then prompt for required information. Once completed, the EXEC will have created the AUDITRE7 and AUDITRE8 modules.

IV.4.2.4 Apply AUDITRE ZAPs

Before proceeding with the AUDITRE installation, apply a ZAP to set the expiration date and apply the CPU authorization. AUDITRE will arrive expired and will need a ZAP to function. There may be some corrections that must be applied to the distributed code. These corrections are supplied as official fixes, which are shipped with the AUDITRE release. Refer to the README dataset for details on current official fixes.

Copy the ZAP statements into a file with fixed-length, 80-byte records, and a filetype of 'ZAP', and then use the command "ZAP MODULE (INPUT filename remove)" to apply them.

IV.5 BS2000 Installation

IV.5.1 Installation Tape

AUDITRE is distributed on a 3490 cartridge. AUDITRE is also available on a small reel of magnetic tape recorded in 9-track, 6250 bpi format with standard labels.

The tape contains two files. The first file is a source file, which contains the AUDITRE install procedure:

```
AUDvrs.INST           (Procedure for Installing AUDITRE)
```

From the second file, a LMS library is constructed with the following members:

```
AUDITRE7              (AUDITRE for ADABAS 7)
AUDITRE8              (AUDITRE for ADABAS 8)
SYSIPT
SAMPLE
```

Note:	Contact Treehouse Software, Inc. for complete Siemens BS2000 instructions.

IV.5.2 Installation Steps

The following steps are required for the BS2000 installation of AUDITRE. It is important to use this list as a guide to be sure that the installation is complete. Each step is covered in detail in this section.

1. Load the AUDvrs.INST procedure.
2. Execute the AUDvrs.INST procedure.
3. Apply AUDITRE ZAPs.
4. Link-edit the AUDITRE Modules.

IV.5.2.1 Load the AUDvrs.INST Procedure

The required space for the LMSLIB is about 1200 PAM-pages.

If the INPL dataset is to be placed onto disk, it requires about 680 PAM-pages.

To install AUDITRE for Siemens BS2000, enter and execute the following procedure, which loads the AUDvrs.INST procedure from the tape:

```
/         PROC    C,(&AVOL=AUDvrs),SUBDTA=&
/         OPTION  MSG=FHL
/         SYSFILE SYSLST=JCL.AUDvrs.INST.PROC
/         REMARK  ***   AUDvrs.INST.PROC   ***
/         ER      AUDvrs.INST
/         STEP
/         SEC     T9G=1
/         FILE    AUDvrs.INST,LINK=EDTSAM,                          -
/                 FCBTYPE=SAM,RECFORM=V,RECSIZE=254,BLKSIZE=2048, -
/                 VOL=&AVOL,DEV=T9G,STATE=FOREIGN
/         SETSW   ON=(4,5)
/         SYSFILE SYSDTA=(SYSCMD)
/         EXEC    $EDT
§0.1
§REA'/FILE'
§SY'REL EDTSAM'
§SY'ER AUDvrs.INST'
§W'AUDvrs.INST'
§H
/         STEP
/         SETSW   OFF=(4,5)
/         SYSFILE SYSDTA=(PRIMARY)
/         SYSFILE SYSLST=(PRIMARY)
/         OPTION  MSG=L
/         ENDP
```

Modify the STEP NATURAL INPL in the AUDvrs.INST procedure.

IV.5.2.2 Execute the AUDvrs.INST Procedure

Execute the AUDvrs.INST procedure, which produces AUDvrs.LMSLIB, an LMS module library (with different types of 'objects').

IV.5.2.3 Apply AUDITRE ZAPs

Before proceeding with the AUDITRE installation, apply a ZAP to set the expiration date and apply the CPU authorization. AUDITRE will arrive expired and will need a ZAP to function. There may be some corrections that must be applied to the distributed code. These corrections are supplied as official fixes, which are shipped with the AUDITRE release. Refer to the README dataset for details on current official fixes.

IV.5.2.4 Link-edit AUDITRE Module

Note:	Please contact Treehouse Software for BS2000 instructions.

IV.6 Operations

IV.6.1 AUDITRE Execution Requirements

Once installation is complete, two programs are available for execution, AUDITRE7 and AUDITRE8. AUDITRE7 can process ADABAS V7.4.x Protection Logs or V8.x protection logs where MU and PE count fields are one byte in length on the PLOG. AUDITRE8 processes ADABAS 8.x Protection Logs where the MU and PE count fields are 1 or 2 bytes in length on the PLOG. When AUDITRE7 is executed, MU and PE occurrence counts occupy one byte in AUDOUT* records; MU and PE occurrence counts occupy three bytes in SHOW or AUDIT reports. AUDITRE8 produces two byte output for MU and PE occurrence counts on AUDOUT* records and five byte occurrence counts in SHOW or AUDIT reports.

Throughout the manual, references to AUDITRE imply AUDITRE7 and AUDITRE8.

AUDITRE requires ADABAS Protection Logging to be turned on with the parameter PROTECTION=YES. If the PLOG input includes spanned block records, the ADABAS parameters creating the PLOG must be defined with SRLOG=ALL to ensure the entire Before and/or After Image is included on the PLOG.

AUDITRE may be run on any CPU whether it runs ADABAS or not, and it does not require NATURAL or any other Software AG product.

Multiple Protection Log datasets are processed by specifying concatenated datasets in the execution JCL.

AUDITRE takes advantage of ADABAS facilities for defining and maintaining logs. Pertinent ADABAS Operations Manual explanations should be consulted for single/dual logging, tape or disk, switching the dual log areas, copying disk logs to tape, etc. The PLOG input to AUDITRE is the output of an ADARES PLCOPY, not the “raw” PLOG that is used by ADABAS.

AUDITRE requires the log records or log datasets to be in chronological order. If records are sorted out of the standard date/time order, AUDITRE will be adversely affected.

IV.6.1.1 AUDITRE Run Time parameters

Specifying an optional run time parameter of PARM=SUMMARY (and DD AUDPRT99 in the Auditre JCL) will generate the end of detail report statistics and the end of summary report statistics, as well as the end of job statistics in AUDPRT99.

To trigger the statistics to print to AUDPRT99 execute Auditre with a SUMMARY parameter specified on the exec statement. The format for the statement is:
//AUD1 EXEC PGM=AUDITREx,PARM='SUMMARY'

In addition, a new DD should be added to the JCL for AUDPRT99, similar to the AUDPRT01 statement.
//AUDPRT99 DD SYSOUT=*

If AUDPRT99 is written to a dataset, it should be defined with LRECL=133, RECFM=FB

IV.6.2 Protection Log Analysis Run

AUDITRE can be used to analyze only the ADABAS Protection Log (PLOG).

Modifications to the standard PLOG record order, format, or field contents prior to processing with AUDITRE should be avoided.

There is no ADABAS standard user-exit facility available for manipulation or reducing PLOG data. Protection Logging can be turned on or turned off for any ADABAS session. Lack of a PLOG for any period of time will not affect AUDITRE, except to make the missing update images unavailable. Lack of a PLOG for any period of time may affect user ability to recover from failures using standard ADABAS facilities.

In order to decompress the "before" and "after" compressed record images on the PLOG, AUDITRE requires a description of the files for which records will be decompressed. This description is obtained from ADACMP (file loader) card images as input to AUDITRE. This description is known as an FDT within AUDITRE.

IV.6.3 Execution Procedure

OS, VM, and SIEMENS

AUDITRE Protection Log processing is executed with the following datasets (Z/OS, VM, or SIEMENS):

Z/OS, VM, or SIEMENS	Meaning
AUDPARM	Input Parameters to AUDITRE.
AUDAPLOG	Input ADABAS Protection Log(s) (tape or disk).
AFDNNNNN	ADABAS FDT (ADACMP) card images describing each file. NNNNN represents the ADABAS FNR
AUDPRT00	Dataset for printing the input parameters, error messages, and execution statistics.
AUDPRT01, 02, 03, etc.	The first dataset is for printing the first detail report and all summary reports. The remaining datasets are for subsequent detail reports.
AUDPRT99	Used to direct the end of detail report statistics and end of summary report statistics, as well as the end of job statistics in AUDPRT99 (requires run time parameter PARM=SUMMARY). (z/OS only)
AUDOUT01, 02, 03, etc.	Output datasets for first, second, etc. output detail logs or summary statistics, allocated with characteristics similar to AUDAPLOG.

Auditre v412 and above allow z/OS sites that do not have a need for the AUDPRT## output to assign the unwanted report files to DUMMY, this may achieve significant CPU savings (dependent on the number of REPORT statements in the AUDPARM – more savings with more REPORT statements). This is useful where the AUDPRT## output is not needed because the site relies on the matching AUDOUT## files for audit purposes. Note: Do not assign AUDPRT00 or AUDPRT99 to DUMMY.

Multiple Input Tapes/Files

For VM, if the INPUT parameter statement specifies TAPES=0, multiple VM/CMS Protection Log tapes or files are processed by operator response to a console message issued by AUDITRE. The message asks if more input tapes are to be processed. A 'Y' (for "YES") instructs AUDITRE to open another file for processing. An 'N' (for "NO") instructs AUDITRE to end the job. If the INPUT parameter statement specifies a fixed number of tapes, exactly that number will be processed by AUDITRE, and no console messages will be issued. (For VM, "TAPES=1" should be specified when processing logs from a disk file.)

ADACMP Cards

ADACMP cards for File 1 must be in dataset AFD00001, File 2 in AFD00002, etc. up to File 65535 in AFD65535. There is a limit of 99 files in any one run.

VSE

AUDITRE Protection Log processing is executed with the following files:

VSE LBL	VSE ASSGN	Meaning
	SYSIPT	Input Parameters to AUDITRE.
AUDAPLG AUDAPLD	SYS010 SYS012	Input ADABAS Protection Log(s) (tape or disk).
AUDFD01, AUDFD02, AUDFD03, AUDFD04, AUDFD05	SYS006, SYS016, SYS026, SYS036, SYS046	Protection Log Analysis ADABAS FDT (ADACMP) card images describing each file. The first file referenced in a SHOW or AUDIT statement is assumed to have ADACMP card images in SYS006, second file in SYS016, etc. up to 5 files. These input files must be FB, LRECL 80, BLKSIZE 80.
	SYSLST	Dataset for printing the input parameters, error messages, and execution statistics.
	SYS005, 015, 025, 035, 045	The first dataset is for printing the first detail report and all summary reports. The remaining datasets are for subsequent detail reports.
AUDOUT1, AUDOUT2, etc.(tape) AUDOT1D, AUDOT2D, etc. (disk)	SYS011, or 013, SYS021, or 023, SYS031, or 033, SYS041, or 043, SYS051, or 053	Output datasets for first, second, etc. output detail logs or summary statistics, allocated with characteristics similar to AUDAPLOG. All output detail records are placed on SYS011 and all output summary records are placed on SYS021, unless OUTPUT statement SYSNO parameters specify otherwise. SYSNO 011, 021, 031, 041, and 051 are for tape output. The remainder are for disk output.

Multiple Input Tapes/Files

If the INPUT parameter statement specifies TAPES=0, multiple VSE Protection Log tapes or files are processed by operator response to a console message issued by AUDITRE. The message asks if more input tapes are to be processed. A 'Y' (for "YES") instructs AUDITRE to open another file for processing. An 'N' (for "NO") instructs AUDITRE to end the job. If the INPUT parameter statement specifies a fixed number of tapes, exactly that number will be processed by AUDITRE, and no console messages will be issued.

ADACMP Cards

ADACMP cards for the first file to be decompressed (the first file referenced in a SHOW or AUDIT statement) must be in SYS006, and the second through fifth files in SYS016, SYS026, SYS036, and SYS046 respectively.

IV.6.4 Z/OS

Sample JCL for a Protection Log Analysis Run
(see source library member JOSPLOG)

```
//AUD     EXEC          PGM=AUDITRE8                                                  (1)
//STEPLIB DD            DSN=AUDITRE.Vvrs.LOAD,DISP=SHR
//AUDAPLOG              DD         DSN=ADABAS.Vx.PROT.LOG,DISP=SHR                    (2)
//AFD00001              DD         DSN=ADABAS.Vx.LOADER(FDT001),DISP=SHR              (2)
//AFD00123              DD         DSN=ADABAS.Vx.LOADER(FDT123),DISP=SHR              (2)
•
• CONTINUE FDT DDS AS NEEDED
•
//AUDPRT00              DD         SYSOUT=A
//AUDPRT01              DD         SYSOUT=A
//AUDPRT02              DD         SYSOUT=A
//AUDPRT03              DD         SYSOUT=A
•
• CONTINUE PRINT DDS AS NEEDED
•
//AUDOUT01              DD         DSN=ADABAS.Vx.PLOGSAVE.XXX,DISP=(OLD,PASS)         (3)
//AUDOUT02              DD         DSN=ADABAS.Vx.PLOGSAVE.YYY,DISP=(OLD,PASS)         (3)
•
• CONTINUE OUTPUT DDS AS NEEDED
•
//AUDPARM               DD         *

         (AUDITRE parameters)
//
```

NOTES:

(1) For ADABAS V7.4.x , use AUDITRE7 . For ADABAS v8.x use AUDITRE8.

(2) The “Vx” in this step should correspond to the ADABAS Version being used (e.g., V813).

IV.6.5 VSE

Sample VSE POWER Execution JCL for a Protection Log Analysis Run With Disk PLOG Input

(see source library member JVSPLOGD)

```
* $$ JOB JNM=AUDITRE,CLASS=Z,DISP=D,PRI=3
* $$ LST CLASS=A           THIS IS FOR SYSLST
* $$ LST LST=AAA,CLASS=Q   WHERE AAA IS A VIRTUAL PRINTER
*
* SAMPLE VSE JCL FOR AUDITRE PROTECTION LOG ANALYSIS
* WITH DISK INPUT
*
// JOB AUDITRE      AUDITRE REPORTS

/* ------------------------------------------------------------------
/* DEPENDING ON THE INPUT PARAMETERS TO AUDITRE, THERE WILL NEED
/* TO BE AT LEAST 1 AND NOT MORE THAN 5 STATEMENTS AS FOLLOWS.
/* THERE SHOULD BE 1 FOR EACH SHOW OR AUDIT STATEMENT SPECIFIED
/* THE INPUT PARAMETERS TO AUDITRE.
/* ------------------------------------------------------------------


/*
// ASSGN SYS005,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 2ND AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS015,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 3RD AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS025,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 4TH AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS035,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 5TH AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS045,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/*
/* BELOW IS THE INPUT DISK PLOG FILE.  REPLACE THE Z'S WITH THE
/* DISK VOLUME, DATASET NAME AND LOCATION ON DISK
/*
// ASSGN SYS012,DISK,VOL=ZZZZZZ,SHR
// DLBL AUDAPLD,'ZZZZZZZ',,SD
// EXTENT SYS012,ZZZZZZ,1,0,ZZZZZ,ZZZ
/*
/* AUDITRE WILL ALLOW UP TO 5 FILES REFERENCED IN THE SHOW OR AUDIT
/* STATEMENTS IN 1 RUN.  AUDITRE REQUIRES THE ADACMP STATEMENTS FOR
/* EACH FILE REFERENCED IN THE SHOW OR AUDIT STATEMENT.  THE 1ST
/* FILE REFERENCED IN A SHOW OR AUDIT STATEMENT IS ASSUMED TO HAVE
/* ADACMP CARD IMAGES IN SYS006, THE 2ND FILE IN SYS016, ETC., WITH
/* THE 5TH FILE IN SYS046.  REPLACE THE X'S WITH THE DISK VOLUME,
/* DATASET NAME AND LOCATION ON DISK OF THE APPROPRIATE ADACMP CARD
/* IMAGE FILES.  THESE FILES MUST BE FB, LRECL 80, BLKSIZE 80.
/*
// ASSGN SYS006,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD01,'XXXXXXX',,SD
// EXTENT SYS006,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 2ND FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS016,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD02,'XXXXXXX',,SD
// EXTENT SYS016,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 3RD FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS026,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD03,'XXXXXXX',,SD
// EXTENT SYS026,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 4TH FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
```

```
// ASSGN SYS036,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD04,'XXXXXXX',,SD
// EXTENT SYS036,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 5TH FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS046,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD05,'XXXXXXX',,SD
// EXTENT SYS046,XXXXXX,1,0,XXXXX,XXX
/*
// LIBDEF *,SEARCH=(AUDLIB.Vvrs)
// LIBDEF PHASE,SEARCH=(AUDLIB.Vvrs)
//* EXEC AUDITRE8,SIZE=1024K   **NOTE IF V7.4 PLOGS EXEC AUDITRE7
                                **NOTE IF V8.x PLOGS EXEC AUDITRE8
* IF DISK INPUT, MUST SPECIFY SYSNO=012 ON INPUT STATEMENT
   INPUT   LOGTYPE=PROTECTION,SYSNO=012
   REPORT     TYPE=DETAIL,HEADING='SHOW UPDATES, FILE 3'
     INCLUDE  FNR=3
     DISPLAY  DATE,TIME,SEQ,RESTART-USERID,FNR,ISN,IMAGE-TYPE
* SHOW REPORT WILL BE IN SYS005, SYS006 SHOULD BE ADACMPS FOR FNR 3
     SHOW     AA,NW,CCC,CC1-5,CG,FNR=3
   REPORT     TYPE=DETAIL,HEADING='AUDIT FILES 1 AND 5'
     INCLUDE  FNR=(1,5)
     DISPLAY  DATE,TIME,SEQ,UIDX,FNR,ISN
* AUDIT REPORT WILL BE IN SYS015, SYS016 SHOULD BE ADACMPS FOR FNR 5
     AUDIT    AA*,NW,CG,CCC,CC1-5,BK,CR,FNR=5
* AUDIT REPORT WILL BE IN SYS025, SYS026 SHOULD BE ADACMPS FOR FNR 1
     AUDIT    AA*,BA*,BB,BC,CB,LA,FNR=1
* ALL SUMMARY REPORTS WILL BE IN SYS005 AFTER ANY DETAIL REPORT
   REPORT     TYPE=SUMMARY,
              HEADING='SUMMARY OF PLOG UPDATES BY FILE/USER'
     CONTROL  FNR,UIDX
/*
/&
* $$ EOJ
```

Sample VSE POWER Execution JCL for a Protection Log Analysis Run With Tape PLOG Input

(see source library member JVSPLOGT)

```
* $$ JOB JNM=AUDITRE,CLASS=Z,DISP=D,PRI=3
* $$ LST CLASS=A           THIS IS FOR SYSLST
* $$ LST LST=AAA,CLASS=Q   WHERE AAA IS A VIRTUAL PRINTER            *
* SAMPLE VSE JCL FOR AUDITRE PROTECTION LOG ANALYSIS
* WITH TAPE INPUT
*
// JOB AUDITRE      AUDITRE REPORTS
/*
/* DEPENDING ON THE INPUT PARAMETERS TO AUDITRE, THERE WILL NEED
/* TO BE AT LEAST 1 AND NOT MORE THAN 5 STATEMENTS AS FOLLOWS.
/* THERE SHOULD BE 1 FOR EACH SHOW OR AUDIT STATEMENT SPECIFIED
/* THE INPUT PARAMETERS TO AUDITRE.
/*

/*
// ASSGN SYS005,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 2ND AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS015,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 3RD AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS025,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 4TH AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS035,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/* STATEMENT BELOW ONLY NEEDED IF 5TH AUDIT OR SHOW STATEMENT IN PARMS
// ASSGN SYS045,AAA        WHERE AAA IS A VIRTUAL PRINTER FROM ABOVE
/*
/* BELOW IS THE INPUT TAPE PLOG FILE.  REPLACE THE Z'S WITH THE
/* TAPE DEVICE.
/*
// ASSGN SYS010,ZZZ
// TLBL AUDAPLG
```

```
/*
/* AUDITRE WILL ALLOW UP TO 5 FILES REFERENCED IN THE SHOW OR AUDIT
/* STATEMENTS IN 1 RUN.  AUDITRE REQUIRES THE ADACMP STATEMENTS FOR
/* EACH FILE REFERENCED IN THE SHOW OR AUDIT STATEMENT.  THE 1ST
/* FILE REFERENCED IN A SHOW OR AUDIT STATEMENT IS ASSUMED TO HAVE
/* ADACMP CARD IMAGES IN SYS006, THE 2ND FILE IN SYS016, ETC., WITH
/* THE 5TH FILE IN SYS046.  REPLACE THE X'S WITH THE DISK VOLUME,
/* DATASET NAME AND LOCATION ON DISK OF THE APPROPRIATE ADACMP CARD
/* IMAGE FILES. THESE INPUT FILES MUST BE FB, LRECL 80, BLKSIZE 80.
/*
// ASSGN SYS006,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD01,'XXXXXXX',,SD
// EXTENT SYS006,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 2ND FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS016,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD02,'XXXXXXX',,SD
// EXTENT SYS016,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 3RD FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS026,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD03,'XXXXXXX',,SD
// EXTENT SYS026,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 4TH FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS036,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD04,'XXXXXXX',,SD
// EXTENT SYS036,XXXXXX,1,0,XXXXX,XXX
/*
/* BELOW 3 LINES ARE ONLY NEEDED IF THERE IS A 5TH FILE SPECIFIED
/* IN THE INPUT PARAMETERS.
/*
// ASSGN SYS046,DISK,VOL=XXXXXX,SHR
// DLBL AUDFD05,'XXXXXXX',,SD
// EXTENT SYS046,XXXXXX,1,0,XXXXX,XXX
/*
// LIBDEF *,SEARCH=(AUDLIB.Vvrs)
// LIBDEF PHASE,SEARCH=(AUDLIB.Vvrs)
// EXEC AUDITRE8,SIZE=1024K   **NOTE IF V7.4 PLOGS EXEC AUDITRE7
*                                **NOTE IF V8.x PLOGS EXEC AUDITRE8
   INPUT   LOGTYPE=PROTECTION
   REPORT     TYPE=DETAIL,HEADING='SHOW UPDATES, FILE 3'
     INCLUDE  FNR=3
     DISPLAY  DATE,TIME,SEQ,RESTART-USERID,FNR,ISN,IMAGE-TYPE
* SHOW REPORT WILL BE IN SYS005, SYS006 SHOULD BE ADACMPS FOR FNR 3
     SHOW     AA,NW,CCC,CC1-5,CG,FNR=3
   REPORT     TYPE=DETAIL,HEADING='AUDIT FILES 1 AND 5'
     INCLUDE  FNR=(1,5)
     DISPLAY  DATE,TIME,SEQ,UIDX,FNR,ISN
* AUDIT REPORT WILL BE IN SYS015, SYS016 SHOULD BE ADACMPS FOR FNR 5
     AUDIT    AA*,NW,CG,CCC,CC1-5,BK,CR,FNR=5
* AUDIT REPORT WILL BE IN SYS025, SYS026 SHOULD BE ADACMPS FOR FNR 1
     AUDIT    AA*,BA*,BB,BC,CB,LA,FNR=1
* ALL SUMMARY REPORTS WILL BE IN SYS005 AFTER ANY DETAIL REPORT
   REPORT     TYPE=SUMMARY,
              HEADING='SUMMARY OF PLOG UPDATES BY FILE/USER'
     CONTROL  FNR,UIDX
/*
/&
* $$ EOJ
```

NOTE:

(1) For ADABAS V7.4.x, use AUDITRE7 . For ADABAS v8.x, use AUDITRE8.

IV.6.6 VM/CMS

Sample VM/CMS EXEC for a Protection Log Analysis Run
(see source file JVMPLOG EXEC)

```
/*   SAMPLE VM EXEC FOR AUDITRE PROTECTION LOG ANALYSIS */

'FILEDEF AUDAPLOG DISK PLOG DATA * (RECFM VB BLOCK 10000'

'FILEDEF AUDPRT00 DISK AUDPLOG LISTING'
'FILEDEF AUDPRT01 DISK AUDREP1 LISTING'
'FILEDEF AUDPRT02 DISK AUDREP2 LISTING'
'FILEDEF AUDPRT03 DISK AUDREP3 LISTING'

/* CONTINUE LISTING FILEDEFS AS NEEDED */


'FILEDEF AUDOUT01 DISK AUDOUT1 OUTPUT A4 (RECFM VB BLOCK 10000'
'FILEDEF AUDOUT02 DISK AUDOUT2 OUTPUT A4 (RECFM VB BLOCK 10000'

/* CONTINUE OUTPUT FILEDEFS AS NEEDED */


'FILEDEF AFD00001 DISK FDT001 ADAWAN *'
'FILEDEF AFD00123 DISK FDT123 ADAWAN *'

/* CONTINUE FDT FILEDEFS AS NEEDED */


'FILEDEF AUDPARM DISK PPLOG AUDPARM *'
/* PPLOG AUDPARM * CONTAINS THE PROTECTION LOG ANALYSIS PARAMETERS */


'AUDITRE8'                                                                    (1)
```

The AUDPLOG PARM file must contain the AUDITRE parameters to use with this execution.

NOTE:
(1) For ADABAS V7.4.x, use AUDITRE7. For ADABAS v8.x, use AUDITRE8.

IV.6.7 BS2000

Sample BS2000 JCL for a Protection Log Run

```
/  PROC    C
/  REMARK
/  REMARK           Sample BS2000 procedure for a
/  REMARK           Protection Log Analysis Run
/  REMARK
/  FILE    ADABASVxx.PLOG,LINK=AUDAPLOG,BLKSIZE=                              (1)
/  FILE    ADABASVxx.FDT001,LINK=AFD00001                                     (1)
/  FILE    ADABASVxx.FDT123,LINK=AFD00123                                     (1)
/  REMARK  continue FDT lines as needed
/  FILE    TTT,LINK=AUDPRT00
/  FILE    UUU,LINK=AUDPRT01
/  FILE    VVV,LINK=AUDPRT02
/  FILE    WWW,LINK=AUDPRT03
/  REMARK  continue print lines as needed
/  FILE    XXX,LINK=AUDOUT01
/  FILE    YYY,LINK=AUDOUT02
/  REMARK  continue output lines as needed
/  SYSFILE TASKLIB=ADA.Vxx.MOD                                                (1)
/  FILE    ZZZ,LINK=AUDPARM           AUDITRE parameter file
/  EXEC    (AUDITRE8,AUDVyyy.LMSLIB)                                          (2)
/  ENDP
```

NOTES:

(1) Supply appropriate ADABAS version library.

(2) For ADABAS V7.4.x, use AUDITRE7 . For ADABAS v8.x, use AUDITRE8.

IV.6.8 Storage Requirements

The storage requirements for running AUDITRE Batch Protection Log facilities are variable. AUDITRE can be run in approximately 1024K bytes of memory if it is to produce only detail reports on a few files and fields and a few simple summary reports.

More virtual storage is required when AUDITRE is expected to produce numerous detail reports on many files and fields or when spanned PLOG records are processed.

Under Z/OS Auditre storage is obtained above the 16MB line when possible. For Z/VSE all storage is obtained from 24-bit GETVIS.

If the user receives an error message indicating that AUDITRE storage limits have been exceeded, an option may be to OUTPUT certain detail log records to a sequential dataset for later AUDITRE processing.

IV.6.9 Processing Time Requirements

The amount of CPU time required for any AUDITRE Protection Log analysis execution is widely variable and depends upon a number of factors, primarily:

- CPU model
- Number of input log records
- Size of input log records
- Number of reports
- Number of input records INCLUDED in reports
- Number of user defined FIELD and VALUE statements
- Number of SHOW and AUDIT fields, especially MU and PE occurrence ranges
- Number of summary reports and number of control fields

IV.6.10 Limits and Restrictions

The AUDITRE7 module is for use with ADABAS 7.4.x and its PLOGs. The AUDITRE8 module is for use with ADABAS 8.x and its PLOGs. Any mismatch of AUDITRE modules with regard to ADABAS versions will cause unpredictable results.

If the PLOG input includes spanned block records, the ADABAS parameters creating the PLOG must be defined with SRLOG=ALL to ensure the entire Before and/or After Image is included on the PLOG.

Every effort has been made to minimize the restrictions and keep AUDITRE limits well beyond commonly reached levels.

The number of input log records is unlimited.

The number of output log records and report print lines is unlimited.

The number of detail reports is limited to 98 for Z/OS, and VM, and 5 for VSE. A typical AUDITRE run may require one to ten detail reports.

The number of summary reports is unlimited (within storage constraints). A typical AUDITRE run may require one to five summary reports.

Each Z/OS and VM report line size can be set to a minimum of 72 characters and a maximum of 4000 characters with 133 the most typical line size. POWER and VSE considerations limit report line size to 121. The page size can be set to a minimum of 10 lines and an unlimited maximum with 55 the most typical page size.

The number of fields that can be stated on a detail report DISPLAY statement is 50 with 4 to 7 being typical. Any field can be stated more than once.

The number of fields that can be stated on a summary report CONTROL statement is 3. This means 3 control levels. One or two control levels is most typical.

The number of "list" values within parentheses for each INCLUDE, EXCLUDE, or VALUE statement is 10. For example: INCLUDE FNR=(1,3,5,7,9,12345) has 6 list values. Multiple INCLUDE, EXCLUDE, and VALUE statements are permitted, which nearly eliminates this restriction.

The number of "derived" fields the user may state via FIELD parameter statements is limited to 40. Each field may have a stated length of a maximum of 16 characters (format = 'C' or 'H') or a maximum of 4 bytes (format = 'B'). Each field may have an unlimited (within storage constraints) number of potential values (VALUE statement).

The number of output sequential datasets (OUTPUT Statement) is limited to 99 for Z/OS and VM.

The number of output sequential datasets (OUTPUT Statement) is limited to 5 for VSE (5 tape and/or disk outputs). VSE outputs can be placed together, eliminating any real restrictions.

The five VSE output datasets are:

- SYS011 (tape), or SYS013 (disk)
- SYS021 (tape), or SYS023 (disk)
- SYS031 (tape), or SYS033 (disk)
- SYS041 (tape), or SYS043 (disk)
- SYS051 (tape), or SYS053 (disk)

A mixture of tape and disk output can be specified for VSE. However, if SYS011 is being used, for example, then SYS013 cannot be used.

The output files for tape have DLBL-names of AUDOUT1, AUDOUT2, AUDOUT3, AUDOUT4 and AUDOUT5.
The output files for disk have DLBL-names of AUDOT1D, AUDOT2D, AUDOT3D, AUDOT4D and AUDOT5D.
Protection Log SHOW and AUDIT fields are limited by a 136K byte output area into which the fields are decompressed. The maximum number of fields which can be stated on one statement is 100. Multiple SHOW and AUDIT statements for the same file can be stated on one report. This causes a concatenation of the fields on the report and output, practically eliminating any real restriction on the number of fields.

SHOW and AUDIT statements cannot be mixed in one report.

The ADACMP input files for VSE must be FB, LRECL 80, BLKSIZE 80.

SECTION V

ERROR DETECTION, PROBLEM DIAGNOSIS

V.1 Error Messages

Error numbers with meaningful messages pertaining to incorrect parameters are usually displayed directly after the parameter in error.

Any parameter error will cause AUDITRE to terminate with condition code 4095 without processing any input log records. In DOS, this termination is via PDUMP with Register 15 containing 4095 (hex FFF). The parameters must be corrected and the job rerun.

Error messages other than for parameter errors will be printed on the applicable report(s).

AUD001 INVALID STATEMENT NAME
The statement name encountered is not a valid AUDITRE statement. Possible causes may include:

- The input parameter file is not an AUDITRE parameter file
- Misspelled or omitted statement name
- Invalid continuation from previous line (missing comma at end of previous line)
- Statement name not followed by a blank

AUD002 INVALID PARAMETER SEPARATOR
The only separators allowed are the comma, greater than symbol, less than symbol, equal sign, and not-equal symbol.

AUD003 NULL PARAMETER INVALID
The parameter cannot be null. A possible cause is a double comma.

AUD004 POSITIONAL PARAMETERS NOT ALLOWED
Either the positional parameter is not allowed, or the maximum of 50 positional parameters was exceeded.

AUD005 UNKNOWN OR INVALID KEYWORD
The keyword parameter is not valid. Possible causes may be a misspelled keyword or a keyword that is valid for another statement but not this one.

AUD006 LIST SIZE EXCEEDED (MAXIMUM 10)
Only 10 entries are allowed in a list of parameters in parentheses.

AUD007 LIST OF VALUES NOT ALLOWED
The parameter must be a single value, not a list of values.

AUD008 REQUIRED PARAMETER MISSING
A required parameter is missing. Possibly it is misspelled, or a comma is missing at the end of the line and the parameter is present on the following line.

AUD009 PARAMETER LENGTH EXCEEDED
The parameter length is larger than permitted.

AUD010 PREMATURE END-OF-FILE
The parameter file has reached the end, but the previous line indicated another line to follow by having a comma at the end of the previous line.

AUD011 INVALID END OR CONTINUATION OF STMT
Parameters must be continued by placing a comma after the last character of the previous line and starting the next line after Column 1. No lines can extend beyond Column 71. Possibly the statement ended before an expected ending quote, a right parenthesis, or a comma was encountered.

AUD012 INVALID NUMERIC VALUE
Only the numeric digits 0 through 9 are allowed for numeric parameters. Possibly an excessive number of digits were specified.

AUD013 INVALID HEXADECIMAL VALUE
Valid hexadecimal digits are 0 through 9 and A through F. Possibly the ending quote is missing.

AUD014 INVALID PARAMETER VALUE
The parameter is not one of the acceptable values.

AUD020 SPANNED RECORD ENCOUNTERED FOR FILE NUMBER xxxxx
AUDITRE will not process ADABAS files defined to allow for records that span blocks. Records from these files will be skipped.

AUD030 STATEMENT NO LONGER VALID
This statement is no longer valid.

AUD031 PARAMETER NO LONGER VALID
This parameter is no longer valid.

AUD040 INPUT STATEMENT REQUIRED
One INPUT statement is required to identify the type of log input being processed. In this version the only valid input logtype is "PROTECTION".

AUD041 INVALID INPUT TYPE
In this version the only valid input logtype is "PROTECTION".

AUD042 STATEMENT INVALID FOR INPUT TYPE
The statements permitted for this input type are listed in **Section III Log Analysis Parameter Statements**.

AUD045 TOO MANY FIELD STATEMENTS
A maximum of 40 user-defined or "derived" fields are allowed.

AUD046 FIELD LENGTH OR DECIMALS INVALID
For 'C' or 'H' format, the field length may be specified as one to 16 bytes of length. For 'B' format, the field length may be specified as one, two, or four bytes of length with one to five decimal positions.

AUD047 FIELD NAME RESERVED OR ALREADY USED
The field statement NAME=name contains a reserved field name (refer to the Expanded Protection Log in Figure 1) or the name was already used in a prior FIELD statement.

AUD048 VALUE STATEMENT BYPASSED
Previous FIELD statement is erroneous, so this VALUE statement is bypassed.

AUD050 REPORT STMT REQUIRED BEFORE THIS STMT
The sequence of statements is invalid. A report statement that identifies the report must precede this statement.

AUD051 INVALID PRINTER LINE-SIZE
The printer line-size is limited to between 72 and 4000 positions.

AUD052 AT LEAST ONE REPORT STATEMENT REQUIRED
At least one report statement must be present. Otherwise, there is no reason to run AUDITRE.

AUD053 DETAIL REPORT EXCEEDS LINE-SIZE
The detail report has been determined to exceed the line-size. Possibly an excessive number of fields are being displayed. The user may define two separate reports to get all the desired fields displayed.

AUD060 INVALID FIELD NAME xxxxxxxx
The indicated field name is not one of the allowable Protection Log fields, one of its alias names, or one of the user-defined fields. Refer to **Section II.2 ADABAS Protection Log Record Fields**.

AUD061 FIELD INAPPROPRIATE FOR SUMMARY REPORT xxxx
The xxxxxxxx field is not one of the numeric log fields possible to perform summary calculations upon, or it is not a numeric user- defined field. Refer to **Section II.2 ADABAS Protection Log Record Fields**.

AUD062 STATEMENT INVALID FOR SUMMARY REPORT
The statement is invalid for a summary report. Possibly, it is valid for a detail report, but not for a summary report.

AUD070 STATEMENT INVALID FOR DETAIL REPORT

The statement is invalid for a detail report. Possibly, it is valid for a summary report, but not for a detail report.

AUD071 CONTROL STMT SUMMARY LEVELS EXCEED 10

The CONTROL statement is limited to 10 fields, providing control levels to a depth of 10. Receiving this error possibly implies a misunderstanding of summary reports. Refer to **Section III.2.9 CONTROL Statement**.

AUD079 REPORT CANNOT HAVE BOTH SHOW AND AUDIT

Change parameter statements to have only SHOW or AUDIT statements for this report.

AUD080 FAILURE GETTING FDT FOR THE FNR

The dataset AFDnnnnn cannot be opened or read. Possibly, the SHOW or AUDIT FNR=nnnnn is an inappropriate file number.

AUD081 NO FDT RECORDS

No FDT records are available for the AFDnnnnn dataset, where nnn is the file number specified on the SHOW or AUDIT statement.

AUD082 SHOW/AUDIT GROUP OR OCCURRING FIELD

The SHOW or AUDIT statement cannot specify a group field or an occurring field without occurrence numbers.

AUD083 SHOW/AUDIT COUNT FOR NON-OCCURRING FIELD

The SHOW or AUDIT statement cannot specify a count field for a non-occurring field.

AUD084 SHOW/AUDIT INVALID FOR LOGTYPE

LOGTYPE must be set to PROTECTION in AUDPARM.

AUD085 SHOW/AUDIT FIELD NOT AVAILABLE

The SHOW or AUDIT field is not a valid field in this file as indicated by the ADACMP cards in the AFDnnnnn dataset.

AUD086 HYPHEN NOT ENCOUNTERED IN RANGE

The SHOW or AUDIT statement specification of an occurring field does not contain a hyphen in what appears to be an occurring range.

AUD087 RANGE ERROR

The SHOW or AUDIT statement occurring field range is invalid.

AUD088 EXCESSIVE SHOW/AUDIT FIELDS

There are too many SHOW or AUDIT fields. A 32K buffer is used to decompress the SHOWn/AUDITed fields. This 32K limit has been exceeded.

AUD089 SHOW/AUDIT MU IN PE SPECIFICATION ERR
There is an error in SHOW or AUDIT specifications for MU within PE (after the #).

AUD099 SEVERE PARAMETER ERROR(S)
An earlier parameter error was detected and a message was printed. The run will be ABENDED with a completion condition code of 4095. No input log records are processed when there are parameter errors.

AUD100 OPEN OPERATION FAILURE ON LOG INPUT
The dataset defined to contain the log input could not be opened. This indicates a JCL specification problem. The run will be aborted with a completion condition code of 4093.

AUD110 OPEN OPERATION FAILURE ON PRINTER nn
The dataset defined to contain the indicated printer-nn output could not be opened. This indicates a JCL specification problem. Printer zero (AUDPRT00) is always required for printing the input parameters. Printer one (AUDPRT01) is always required to print any reports. Printer two, three, etc., are required if and only if there are multiple detail reports. Possibly there are not enough printer datasets defined for the number of detail reports requested. The run will be aborted with a completion condition code of 4092.

AUD111 OUTPUT ERROR ON PRINTER nn
The dataset defined to contain the indicated printer-nn output has malfunctioned. The run will not be aborted. The missing report can be run at a later time.

AUD120 OPEN OPERATION FAILURE ON OUTPUT nn
The nnth dataset defined to contain detail or summary output could not be opened. This indicates a JCL specification problem. The run will be aborted with a completion condition code of 4091.

AUD121 OUTPUT ERROR ON OUTPUT nn
The nnth dataset defined to contain the detail or summary output has malfunctioned. The run will not be aborted. The missing output can be generated at a later time.

AUD190 CLOSE OPERATION FAILURE ON xxxxxxxx
The xxxxxxxx dataset close operation has failed. The run will not be aborted. If the file is an output file, the data may have to be regenerated.

AUD200 REPORT LIMIT REACHED
This warning indicates the limit of records specified for this report has been reached. This as may have been a test run to view the type of report being output. If not, possibly the limit should be raised or removed altogether for subsequent runs.

AUD201 SUMMARY REPORT EXCEEDS LINE SIZE
The exact format of the summary report is not determined until all input log records are processed. Only then are the values of all sums, maximums, averages, etc., known. Only then can the report be formatted to a logical columnar design. All fields up to the one which goes beyond the line-size will be printed. For subsequent runs, either:

- Define a larger line-size, if possible;

- Specify less summary calculations on this report; or
- Define another report to contain additional necessary summary figures.

V.2 Condition Codes

Upon successful completion in OS and VM, AUDITRE will set the condition code (Register 15) to zero before returning to the caller (operating system). In DOS, AUDITRE will execute an EOJ.

Abnormal termination in OS will result in the setting of the following condition codes:

CODE	MEANING
4095	A parameter error has been encountered. Refer to the report on printer zero (AUDPRT00). At least one error message will identify the error. All parameter errors must be corrected before AUDITRE will process input log records.
4094	Insufficient storage to process desired reports.
4093	An error has been encountered in attempting to open or read the input parameter file. Check the JCL.
4092	An error has been encountered in attempting to open the input log file. Check the JCL.
4091	An error has been encountered in attempting to open an output dataset. The output dataset number (01-99) is noted on an error message on printer zero.
4090	An error has been encountered in attempting to open printer zero (AUDPRT00) or a report printer dataset. The printer number (01-99) is noted on an error message on printer zero.
4087	Failure in attempting to open or read the FDT (ADACMP dataset) for the SHOW or AUDIT FNR.

In DOS, abnormal termination will be via PDUMP with Register 15 containing one of the above codes.

V.3 Frequently Encountered Problems

AUDITRE MODULE/LIBRARY MANAGEMENT

Each new version of AUDITRE is distributed under a new library name (AUDITRE.Vvrs.LOAD). Since the member names may stay the same from version to version, it is **IMPORTANT** to verify that the new version of AUDITRE is using the new members that were provided on the tape. Upon installing this library, users often rename, copy, move members, etc. Fixes (zaps) supplied by Treehouse Software often get applied without resulting in any apparent fix of the problem because the wrong library or wrong member has been zapped.

COMMAND SEQUENCE OR PARAMETER SYNTAX ERROR MESSAGES

Parameter error messages, incorrectly processed parameters, and unexpected results can be avoided by using proper Command sequencing and Parameter syntax.

OUTPUT FILE APPEARS INCOMPLETE

Output was limited by INPUT LIMIT=, INCLUDE/EXCLUDE, or OUTPUT LIMIT= statements.

PROCESSING ABENDS WHEN LOADER DEFINITION CARD IMAGE DATASET (FILE) IS READ

The DDCARD (File Label) for the Loader definition dataset (file) may be missing from the Job Control.

Possibly the loader dataset (file) definition is not in 80 character format (unblocked for DOS).

PROCESSING PROBLEM (DOS)

When processing multiple ADABAS Files in a single PLOG run, reference the Files in the same sequence as their Loader definition files are defined in the JCL (SYS006 for first file, SYS016 for second file, etc).

REPORT APPEARS INCOMPLETE

The report may have been limited by INPUT LIMIT=, REPORT LIMIT=, or INCLUDE/EXCLUDE statements.

REPORT CONTAINS DATA THOUGHT TO BE EXCLUDED

Refer to the description and examples for the INCLUDE/EXCLUDE statements in the Reference Manual.

TIME IS INACCURATE BY A MULTIPLE OF HOURS

Local CPU is IPLed with other than local time. Use the INPUT CLOCK-FACTOR= parameter to adjust to local time.

AUDITRE DOES NOT PRODUCE AN EXPECTED REPORT, AND THERE IS NO ERROR MESSAGE

One of the following conditions is true:

- No records were selected by the INCLUDE statement(s), or all records were rejected by the EXCLUDE statement(s).

- No records were found for this report within the set of records processed by the INPUT LIMIT= statement.

- Input tape (DOS) has a bad record or is missing a label record.

- There was a mismatch between the AUDITRE version and the PLOG ADABAS version.

USER DEFINED FIELD PROBLEM

User defined fields must not use a FIELD name already being used by AUDITRE. For example, the user defined field cannot be named "FNR".

PROCESSING IMPOSSIBLE DUE TO SHORTAGE OF MEMORY

Too many reports were requested, or too many fields were SHOWn or AUDITed.

AUD014 WITH ABEND, BUT STATEMENTS APPEAR TO BE OKAY

Parameters must be in the correct sequence.

Too many fields may have been defined in a single DISPLAY, SHOW, or AUDIT statement.

V.4 Frequently Asked Questions

Q. The printout produced by AUDITRE displays the changed fields one per line, and all of the BEFORE field values appear before the AFTER field values. I see that I can select the fields to be shown, but sometimes all of the fields are not printed. I want to write a program to further analyze the changed fields, and this format is unwieldy for my purposes. Do you have any suggestions?

A. The short answer is to write your program to process the AUDITRE "OUTPUT data", not the report data. An explanation follows.

For each file you decide to look at, AUDITRE does allow you to select the fields to be shown (SHOW statement) or the fields to have their before and after images compared for change (AUDIT statement). In either case, the fields are first decompressed and strung out in exactly the FDT length and format into what we call a "potential output buffer".

If your FDT states that the file has fields AA, BB, ... ZZ, and your SHOW statement or AUDIT statement lists AA, CC, EE, GG, then the output buffer will have (after an 80 byte header containing the date, time, file, ISN, Before/After indicator, etc.) exactly fields AA, CC, EE, and GG. It does not matter if any of these fields were updated, added, or deleted. It does not matter if AUDITRE sees a Before or an After image. Every record (image) selected for this file will have exactly this length and format of data. Let's say the fields are 5, 6, 7, and 8 bytes, respectively. Then, the output buffer is 80 + 26 = 106 bytes for this file's output buffer. If you have another report with another set of fields for this file, then AUDITRE would make another potential output buffer with the different fields' lengths and format.

We term this a potential output buffer because it will only be output if you have an OUTPUT statement, and if you have the parameter COMPRESS=NO. If you have COMPRESS=YES, then the output will look like the input (i.e., compressed).

So, your output buffer for some updates will appear as:

header for a Before image	AA value	CC	EE	GG
header for an After image	AA value	CC	EE	GG
header for a Before image	AA value	CC	EE	GG
header for an After image	AA value	CC	EE	GG

etc.

This buffer format is the same for each and every record for this file for this SHOW or AUDIT statement. Some people ask, "if only field AA got updated, will the output have only field AA?" No, the output will have fields AA, CC, EE, and GG. The user would not be able to easily process the data that AUDITRE placed in the output buffer if the data appeared randomly as field AA, or CC, or ZZ, etc. If the output buffer with some kind of header was 85 bytes, would this include the 5-byte AA, or would it be YY and ZZ that are 2 and 3 bytes? The user would never be able to figure it out unless AUDITRE would output the FDT along with the data, and mark in some way each field value as being for a certain field. That would be a big mess. So instead, all of the listed fields have to be there all the time, giving the output a "flat file" format. The user can predict the format and build a COBOL or NATURAL input data area very easily.

What gets printed on your detail lines is one field per line, unless the field is so big that it takes up 2 or 3 lines. If you have a SHOW statement, you will see:

AA value (may be edited or decompressed)
CC value
EE value
GG value

If it was an AUDIT statement, the example above was a Before and After image for the same ISN (an update scenario), and only AA, CC, and GG had their values changed, then you would see:

A:AA value
B:AA value

A:CC value
B:CC value

A:GG value
B:GG value

In other words, there will be one (in the case of an add or delete) or two (in the case of an update) lines per field on the detail report in a very readable report. However, there will be one or two records in the OUTPUT dataset, which is much easier to "post process" with some NATURAL or COBOL program.

The Before and After values of one updated field will not be together in one output record. They would be in two records, along with all the other fields stated in the AUDIT statement, as described earlier.

Q. Does AUDITRE unpack all numeric fields?

A. When AUDITRE outputs (OUTPUT statement) a record, it does not necessarily unpack fields. AUDITRE does unpack a field whose format is stated in the supplied FDT to be unpacked because ADABAS packs all unpacked data on the database, so the PLOG has it packed, and it clearly should be unpacked to be presented to the user. However, AUDITRE does not unpack a field whose format is packed, and the ADABAS PLOG has as packed because it clearly should stay packed if the FDT states that that is how the user requested it. This is information about the output buffer.

For detail reports, AUDITRE does unpack any packed field when it puts it on the report line. This is because AUDITRE thinks its users would rather see 15,482 than x'3c7a'. Furthermore, AUDITRE edits an unpacked field because AUDITRE thinks the user would rather see 12,345 as opposed to 1234N. A binary field is displayed in hex and in decimal rather than as 10101010101010101.

V.5 Problem Diagnosis Checklists

If none of the “Common Problems” are the cause of an AUDITRE batch reporting problem, consider the following checklist:

- Is the user executing an AUDITRE module of the same version in which the PLOGs were generated?
  AUDITRE7 is for use with ADABAS 7.4.x PLOGs only. AUDITRE8 is for use with ADABAS 8.x PLOGs only.

- Did this problem exist in prior versions of AUDITRE?
  If this is the case, contact Treehouse Software.

- Are all TSI official zaps applied? Are they applied correctly?
  The failure to apply all official zaps or the incorrect application of an official zap can have a severe negative impact on the operation of AUDITRE. Verify all zaps. If the problem persists, contact Treehouse Software. At that time, verification can be made that all official zaps have been received.

- Was a large enough region supplied in which to run the report?
  The lack of sufficient storage area can cause batch reporting to ABEND. Try a larger region size before calling Treehouse Software.

- What specifically is wrong with the report?
  When contacting Treehouse Software, specify precisely what is wrong with the report (i.e., it contains invalid data, the headings are wrong, etc.).

- Does the JCL point to the correct Load Library?
  When new versions of AUDITRE are installed, customers sometimes fail to update JCL to reflect the new Load Libraries.

- Did the correct “FNR=nnnnn” appear on the SHOW or AUDIT statement?
  The "FNR=nnnnn" parameter states the ADABAS file number. If this parameter is missing, file zero will be assumed, ending the run as a JCL error for lack of ADACMP card images for file zero.

- Are file definitions correct?
  Protection Log problems are almost always due to the file definitions not matching what is actually on the file. Verify that the definitions are correct by dumping the Protection Log (or run an AUDITRE report including the ABENDing record and DISPLAY IMAGE) and compare it with the ADACMP file definition.

  Usual problems that would cause an abend during PLOG processing are:

  - *The PLOG being processed is from the wrong database, but it happens to have the same file number. (i.e., There are different File 102s with different field layouts on databases 250 and 251.)*

  - *The ADACMP definitions and the ADABAS internal FDT do not match. It may be that the file definition was recently modified and the AUDITRE job being run is using the new file definition with an old PLOG tape. Upon closer investigation, the customer will probably discover that this happens on one particular record in a particular file, probably the first record for the file.*

- Has a PLOG "phased-test" been tried?
  When a complicated PLOG run ABENDs, try running one PLOG, one Report, on one file, DISPLAYing some "fixed fields", but SHOWing no ADABAS fields. If this works, try SHOWing the first field, the second field, etc. Do not SHOW ALL. Eventually phase in the more complex AUDIT statement. Steadily increase complexity until the failure occurs.

  To determine the problem, TSI (or the customer) will need the following items:

  - *ADACMP dataset supplied to AUDITRE*
  - *AUDITRE parameters used to process the file*
  - *AUDITRE report (several pages up to the failure point)*
  - *A "dump" of the PLOG for the failing record*

The PLOG "dump" can be accomplished in the following manner:

- *If the AUDITRE parameters report on multiple files, then temporarily reduce the complexity of the run to the failing file. Then reduce the run to the failing record. There may already be enough information to do so, such as a SEQ or TIME field value, which can be used in an INCLUDE for the next run.*
- *Process the PLOG with a DISPLAY IMAGE statement included with the rest of the parameters. This DISPLAY IMAGE can be placed after any other DISPLAY statements and before any SHOW statements. The DISPLAY IMAGE will cause a hex dump of the entire compressed record that AUDITRE is attempting to decompress.*
- *If the DISPLAY IMAGE results in a huge printout then try running against the PLOG one time without the DISPLAY IMAGE for the purpose of determining the failing record. DISPLAY the SEQ, TIME, FNR, etc. fields. Then run against the PLOG again, "dumping" a small portion of records that include the failing record with a DISPLAY of the SEQ, TIME, FNR, and IMAGE fields.*

After acquiring the information above, often the problem can be seen if one understands the rules for decompression. If not, TSI should be able to help. As a last resort, send the collected information to TSI.

APPENDIX A

GENERAL PARAMETER RULES

All batch AUDITRE runs are controlled by parameter statements.

Each parameter statement consists of an op-code and one or more operands. The parameters follow the general rules for assembler coding:

- Op-code must start after column 1 and must be followed by at least one blank
- Operands are positional or key-word, follow the op-code, are separated by commas, can be coded up to column 71, and must be followed by at least one blank
- Multiple operands can be spread over multiple lines
- Comments can follow operands or can be indicated by an asterisk in column 1
- No imbedded blanks are allowed within the op-code and operands except within apostrophes

This page intentionally left blank.

APPENDIX B

OUTPUT DATASET FORMATS

The format of output "COMPRESS=NO" Protection Log datasets for Detail Reports is as follows:

- Record length, 2 bytes binary, followed by 2 bytes of X'0000'
- Record "header", 80 bytes
  - "AUDvrs", 6 bytes
  - "AP", 2 bytes meaning ADABAS Protection Log output
  - Date of AUDITRE run, 8 bytes YY-MM-DD for YEARFMT=2, YYYYMMDD for YEARFMT=4
  - Time of AUDITRE run, 8 bytes HH:MM:SS,
  - Date (YY-MM-DD for YEARFMT=2, YYYYMMDD for YEARFMT=4) and time from input Protection Log, 16 bytes
  - The "ID" of the record, 8 bytes. (This ID is the value stated for the ID parameter on the OUTPUT statement.)
  - DBID, 2 bytes binary (DBID 1-65535)
  - File Number, 2 bytes binary
  - ISN, 4 bytes binary
  - Image-type, 1 byte (“B” for Before, “A” for After)
  - Restart Userid, 8 bytes
  - Userid8, 8 bytes
  - Modification Type (AUDIT Statement only)
    Update: "U"
    Add: "A"
    Delete: "D"
  - Transaction Sequence Number, 4 bytes
  - unused, 2 bytes or the two byte spanned record segment indicator (see “Spanned Record Layout and Processing” below)
- Data portion, length depends upon field lengths
  - First SHOWn/AUDITed field value. Length and format depends upon field length and format as stated in the file description (ADACMP).
  - Second SHOWn/AUDITed field value.
  - Third SHOWn/AUDITed field value.
  - Etc.

Following is an example related to the “COMPRESS=NO” dataset data portion as described on the previous page.

Assume the file description (ADACMP) for a file is as follows:

```
ADACMP FNDEF='01,AA,06,A'                simple elementary field
ADACMP FNDEF='01,BB,04,A,MU(5)'          MU field
ADACMP FNDEF='01,CC,PE(4)'               PE group
ADACMP FNDEF='02,CP,2,A,NU'
ADACMP FNDEF='02,CQ,2,P,NU'
ADACMP FNDEF='02,CR,2,U,NU'
ADACMP FNDEF='02,CS,2,A,NU,MU(3)'        MU field within PE group
ADACMP FNDEF='02,CT,2,B,NU'
```

Assume the first PLOG image, which is decompressed by AUDITRE and is to be placed into the output dataset contains:

For field AA: ABCDE

For field BB: three occurrences with the values ZZZZ, YYY, X

For field CC: two occurrences with the values:

CP: OH, MY
CQ: x'1F', x'123F'
CR: null, x'045F
CS #1: two occurrences with the values:
ZZ,YY
CS #2: one occurrence with the value:
AA
CT: x'5555', null

Assume an AUDIT or SHOW statement as follows:

```
AUDIT AA,BB2-3,CCC,CQ2,CR1-3
```

The output file data portion representing this first PLOG image would contain:

```
x'C1C2C3C4C540E8E8E840E740404002123F000F045F000F'
```

Note that subsequent PLOG images would have the same fields' values decompressed and placed in exactly the same positions in the output record. Every output record in this example would have the same length and field positioning in this data portion. The length of this data portion would always be 23 bytes for ADABAS 7.4.x, i.e., the length of field AA, two times the length of BB, one (the one-byte count field, CCC), the length of CQ, and three times the length of CR. The length of this data portion would always be 24 bytes for ADABAS v8.x, i.e., the length of field AA, two times the length of BB, two (the two-byte count field, CCC), the length of CQ, and three times the length of CR.

Note that the above example SHOWs or AUDITs a variety of fields in the file, and these fields are appropriately placed in the OUTPUT dataset.

In the case of SHOW ALL or AUDIT ALL, AUDITRE assumes the user desires to output every elementary field and every occurrence of MU fields and PE fields within each PE group in FNDEF field order. However, it is usually not reasonable to include all 191 maximum occurrences for all MU fields and all 191 occurrences for all PE groups for ADABAS 7.4.x. These maximum MU and PE occurrence number can be 65,535 for ADABAS 8.x. AUDITRE takes care of this situation as described on the next page.

AUDITRE is intending to align each record in the output dataset so that each field has a predictable location in the data portion, just as AUDITRE would do in the previous SHOW/AUDIT example, in which the user stated specific fields. AUDITRE is guided in the placement of these fields based upon the field lengths and the occurrence numbers for the MU fields and PE groups. These occurrence numbers are not the value in the count field (which can vary in each image), but are the values as stated in the SHOW/AUDIT statement. In the previous example, the AUDIT statement stated the user's desire to output two occurrences of values in field BB, regardless of the fact that the various PLOG images may have anywhere from zero occurrences to 191. In that example, field BB had three occurrences, with the first occurrence not desired (BB2-3).

In the case of SHOW ALL or AUDIT ALL, AUDITRE will not assume maximum occurrences for all MU fields and PE groups are desired (this is usually unreasonable). AUDITRE will instead assume 10 occurrences are desired, unless the user includes some other number of occurrences to be considered. This is done by placing some number from 1 to 191 for ADABAS v7.4.x and 1 to 65535 for ADABAS v8.x for MU fields or PE groups in the ADACMP FNDEF statement after the MU or PE notation. This number is placed in parentheses. Note in the previous example, field BB is stated to have a reasonable limit of 5 occurrences, field CC is stated to have a reasonable limit of 4 occurrences, and field CS is stated to have a reasonable limit of 3 occurrences.

Also in the case of SHOW ALL or AUDIT ALL, AUDITRE will assume the user desires to see all MU and PE counts of occurrences. This one-byte binary count of the number of occurrences found in the PLOG image will precede the first MU value or the first value of the first field within the PE group as appropriate. In the case of an MU within a PE, the one-byte MU count will precede each occurrence within the PE.

Following is a SHOW/AUDIT ALL example of the output dataset data portion for the single PLOG image shown in the example above for ADABAS v7.4.x:

```
x'C1C2C3C4C54003E9E9E9E9E8E8E840E74040404040404040404002D6C8D4E840404040001F123F
000F000FF0F0F4F5F0F0F0F002E9E9E8E8404001C1C14040404000404040404040004040404040405
55000000000000'
```

An explanation of the hex data above follows:

x' C1C2C3C4C540'	The value for the field AA, ABCDE followed by a blank, for a total of 6 bytes.
x'03'	The number of occurrences of BB found in the PLOG image. This count is included prior to the first field value since SHOW ALL or AUDIT ALL is specified.
x' E9E9E9E9E8E8E840E74040404040404040404040'	Five occurrences of BB (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 5) with a 4-byte alphanumeric value as defined in the ADACMP statements. Note the space filling on the right of some values and four spaces for any occurrences that do not exist.
x'02'	The number of occurrences of PE group CC found in the PLOG image. This count is included prior to the first field value since SHOW ALL or AUDIT ALL is specified.
x'D6C8D4E840404040'	Four occurrences of CP (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.

x'001F123F000F000F'	Four occurrences of CQ (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte packed value as defined in the ADACMP statements. Note the x'000F' for any occurrences that do not exist.
x'F0F0F4F5F0F0F0F0'	Four occurrences of CR (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte unpacked value as defined in the ADACMP statements. Note the x'F0F0' for any occurrences that do not exist.
X'02'	The number of occurrences of CS #1 found in the PLOG.
X'E9E9E8E84040'	Three occurrences of CS#1 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'01'	The number of occurrences of CS #2 found in the PLOG.
X'C1C140404040'	Three occurrences of CS#2 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'00'	The number of occurrences of CS #3 found in the PLOG.
X'404040404040'	Three occurrences of CS#3 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'00'	The number of occurrences of CS #4 found in the PLOG.
X'404040404040'	Three occurrences of CS#4 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
x'5555000000000000'	Four occurrences of CT (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte binary value as defined in the ADACMP statements. Note the x'0000' for any occurrences that do not exist.

Following is a SHOW/AUDIT ALL example of the output dataset data portion for the single PLOG image shown in the example above for ADABAS v8.x:

```
x'C1C2C3C4C5400003E9E9E9E9E8E8E840E74040404040404040404040400002D6C8D4E840404040001F
123F000F000FF0F0F4F5F0F0F0F00002E9E9E8E840400001C1C1404040400000404040404040000040
40404040405555000000000000'
```

An explanation of the hex data above follows:

x' C1C2C3C4C540'	The value for the field AA, ABCDE followed by a blank, for a total of 6 bytes.
x'0003'	The number of occurrences of BB found in the PLOG image. This count is included prior to the first field value since SHOW ALL or AUDIT ALL is specified.
x' E9E9E9E9E8E8E840E74040404040404040404040'	Five occurrences of BB (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 5) with a 4-byte alphanumeric value as defined in the ADACMP statements. Note the space filling on the right of some values and four spaces for any occurrences that do not exist.
x'0002'	The number of occurrences of PE group CC found in the PLOG image. This count is included prior to the first field value since SHOW ALL or AUDIT ALL is specified.
x'D6C8D4E840404040'	Four occurrences of CP (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.

x'001F123F000F000F'	Four occurrences of CQ (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte packed value as defined in the ADACMP statements. Note the x'000F' for any occurrences that do not exist.
x'F0F0F4F5F0F0F0F0'	Four occurrences of CR (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte unpacked value as defined in the ADACMP statements. Note the x'F0F0' for any occurrences that do not exist.
X'0002'	The number of occurrences of CS #1 found in the PLOG.
X'E9E9E8E84040'	Three occurrences of CS#1 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'0001'	The number of occurrences of CS #2 found in the PLOG.
X'C1C140404040'	Three occurrences of CS#2 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'0000'	The number of occurrences of CS #3 found in the PLOG.
X'404040404040'	Three occurrences of CS#3 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
X'0000'	The number of occurrences of CS #4 found in the PLOG.
X'404040404040'	Three occurrences of CS#4 (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for this MU was 3) with a 2-byte alphanumeric value as defined in the ADACMP statements. Note the spaces for any occurrences that do not exist.
x'5555000000000000'	Four occurrences of CT (because SHOW ALL or AUDIT ALL was specified and the number of occurrences indicated in the ADACMP definition for the PE group CC was 4) with a 2-byte binary value as defined in the ADACMP statements. Note the x'0000' for any occurrences that do not exist.

Note that subsequent PLOG images for this SHOW/AUDIT example would have the same fields' values decompressed and placed in exactly the same positions in the output record. Every output record in this example would have the same length and field positioning in this data portion.

The length of this data portion would always be 88 bytes for ADABAS v7.4.x (i.e., the length of field AA, one (the one-byte count field, BBC), five times the length of BB, one (the one-byte count field, CCC), four times the length of CP, four times the length of CQ, four times the length of CR, one (the one-byte count field for CS occurrence number 1), followed by three times the length of CS for CS PE occurrence number 1, one (the one-byte count field for CS occurrence number 2), followed by three times the length of CS for CS PE occurrence number 2, one (the one-byte count field for CS occurrence number 3), followed by three times the length of CS for CS PE occurrence number 3, one (the one-byte count field for CS occurrence number 4), followed by three times the length of CS for CS PE occurrence number 4, and four times the length of CT).

The length of this data portion would always be 94 bytes for ADABAS v8.x (i.e., the length of field AA, two (the two-byte count field, BBC), five times the length of BB, two (the two-byte count field, CCC), four times the length of CP, four times the length of CQ, four times the length of CR, two (the two-byte count field for CS occurrence number 1), followed by three times the length of CS for CS PE occurrence number 1, two (the two-byte count field for CS occurrence number 2), followed by three times the length of CS for CS PE occurrence number 2, two (the two-byte count field for CS occurrence number 3), followed by three times the length of CS for CS PE occurrence number 3, two (the two-byte count field for CS occurrence number 4), followed by three times the length of CS for CS PE occurrence number 4, and four times the length of CT).

The format of output "COMPRESS=YES" Protection Log datasets for Detail Reports is almost identical to the input Protection Log format, so these datasets can be processed further by AUDITRE. However, do not attempt to use AUDITRE outputs as input to any ADABAS utilities.

The output "COMPRESS=YES" Protection Log datasets for Detail Reports includes ONLY before/after image records. Separator records, FDT-change records, user-records, or other non-before/after image records from the input PLOG will not occur in the output "COMPRESS=YES" Protection Log datasets for Detail Reports.

The format of output Summary datasets is as follows:

- Record length, 2 bytes binary, followed by 2 bytes of X'0000'
- Record "header", 80 bytes
  - "AUDvrs", 6 bytes
  - "AP", 2 bytes meaning ADABAS Protection Log output,
  - Date of AUDITRE run, 8 bytes YY-MM-DD for YEARFMT=2, YYYYMMDD for YEARFMT=4
  - Time of AUDITRE run, 8 bytes HH:MM:SS,
  - Earliest date (YY-MM-DD for YEARFMT=2, YYYYMMDD for YEARFMT=4) and time from input Protection Log, 16 bytes
  - Latest date (YY-MM-DD for YEARFMT=2, YYYMMDD for YEARFMT=4) and time from input Protection Log, 16 bytes
  - "ID" of the record, 8 bytes. (This is the value stated for the ID parameter on the OUTPUT statement.)
  - Unused, 16 bytes
- Control portion
  - First CONTROL field value; length and format depends upon field length and format (See the Sections on ADABAS Protection Log Fields.)
  - Second CONTROL field value
  - etc., up to 3 CONTROL fields maximum
- Record count for CONTROL field(s), 4 bytes maximum and aligned on a full-word boundary

Spanned Record Layout and Processing

Auditre version 5.x and above can process Adabas PLOG data containing spanned records. Adabas version 8.2 and above can write spanned record segments to the Adabas protection log. Up to five segments may be written by Adabas. The number of segments in the Auditre output file depends on the size of the decompressed fields reported and the logical record length of the output report dataset. The maximum logical record size supported is 32,756, with a maximum blocksize of 32,760. If the PLOG input includes spanned block records, the ADABAS parameters creating the PLOG must be defined with SRLOG=ALL to ensure the entire Before and/or After Image is included on the PLOG.

To support spanned records, when generating OUTPUT COMPRESS=NO data the last two bytes of the Auditre header are replaced with a byte indicating the segment type, primary or secondary, followed by a one byte binary count field. A primary segment is indicated by a 'P' (X'D7') in the indicator byte followed by the number of secondary segment records to follow. Each secondary segment is indicated by a 'S' (X'E2') in the indicator byte followed by a single binary byte indicating the segment count. Thus if five spanned PLOG segments is produced by Auditre, the primary segment will show X'D704'. The secondary segments will show X'E201', 'XE202', X'E203' and X'E204' respectively in the four secondary segment records that follow.

When a set of spanned record segments are written Adabas field boundaries are not respected. Thus, the program processing the output data must combine the segments in storage to build the complete record for processing.

The last two bytes of the header will contain two blanks if the output record was derived from a non-spanned input PLOG record.

Spanned records processed with OUTPUT COMPRESS=YES are written from their original PLOG segments with the Auditre header replacing the Adabas PLOG header. A subsequent Auditre run to process these records can logically combine the spanned segments for show and audit reports and for OUTPUT COMPRESS=NO output.

APPENDIX C

SAMPLE PROTECTION LOG ANALYSIS INPUT PARAMETERS AND REPORTS

Assume that a short NATURAL session produces an ADABAS Protection Log of updates. Two programs, UPDPERS and FINANC are coded, stowed, and executed. UPDPERS updates AGE and HOBBY fields on 4 PERSONNEL file records. FINANC updates NET-WORTH and COLLEGE fields on 3 FINANCE file records.

It would be useful to see AA (PERSONNEL-NUMBER) for those persons whose NW (NET-WORTH) changed.

The SHOW statement could be used for limited analysis of this log. Consider the following example of the SHOW statement:

```
INPUT   LOGTYPE=PROTECTION
REPORT  TYPE=DETAIL,HEADING='SHOW UPDATES, FILE 3'
INCLUDE FNR=3
DISPLAY DATE,TIME,SEQ,RESTART-USERID,FNR,ISN,IMAGE-TYPE
SHOW    AA,NW,CCC,CC1-5,CG,FNR=3
```

The output generated is:

```
SHOW UPDATES, FILE 3        AUDITRE 5.2          MON 14-01-31 17:45:36

YY-DDD   HH-MM-SS    SEQ    RES-UID    FNR       ISN    IMAGTYP
14-031 15:03:17       29    TREE2        3         4    BEFORE

AA= HEX 000000000186F4  DEC 100,084                              PERSONNEL-NUMBER
NW=            1111                                             NET-WORTH
CCC=  1                                                         CREDIT-CARD
CC  1=BANK AMERICARD                                            CREDIT-CARD
CC  2=                                                          CREDIT-CARD
CC  3=                                                          CREDIT-CARD
CC  4=                                                          CREDIT-CARD
CC  5=                                                          CREDIT-CARD
CG=GROVE CITY                                                   COLLEGE

14-031   15:03:17     30    TREE2        3         4    AFTER

AA= HEX 000000000186F4  DEC 100,084                              PERSONNEL-NUMBER
NW=            2222                                             NET-WORTH
CCC=  1                                                         CREDIT-CARD
CC  1=BANK AMERICARD                                            CREDIT-CARD
CC  2=                                                          CREDIT-CARD
CC  3=                                                          CREDIT-CARD
CC  4=                                                          CREDIT-CARD
CC  5=                                                          CREDIT-CARD
CG=SLIPPERY ROCK                                                COLLEGE
```

```
14-031   15:03:19     31    TREE2        3          5    BEFORE

AA= HEX 000000000186F5  DEC 100,085                              PERSONNEL-NUMBER
NW=            3333                                              NET-WORTH
CCC=  2                                                          CREDIT-CARD
CC  1=DINERS CLUB                                                CREDIT-CARD
CC  2=AMERICAN EXPRESS                                           CREDIT-CARD
CC  3=                                                           CREDIT-CARD
CC  4=                                                           CREDIT-CARD
CC  5=                                                           CREDIT-CARD
CG=GROVE CITY                                                    COLLEGE

14-031   15:03:19     32    TREE2        3          5    AFTER

AA= HEX 000000000186F5  DEC 100,085                              PERSONNEL-NUMBER
NW=            4444                                              NET-WORTH
CCC=  1                                                          CREDIT-CARD
CC  1=DINERS CLUB                                                CREDIT-CARD
CC  2=AMERICAN EXPRESS                                           CREDIT-CARD
CC  4=                                                           CREDIT-CARD
CC  5=                                                           CREDIT-CARD
CG=SLIPPERY ROCK                                                 COLLEGE
```

The type of printout above would be repeated for four more records (two more updates) with a final tally:

```
NUMBER OF INPUT RECORDS:     36
INPUT RECORDS INCLUDED:       8
```

Of course, multiple PLOGs can be processed in one pass to generate multiple reports. Each report can show the contents of one or more files and the contents of selected fields from selected records. The data could be output to a 'flat file' decompressed form of dataset by coding the following parameter:

```
       OUTPUT ID=F3UPD
```

This will result in flat file records having an 80-character standard 'header' identifying DATE, TIME, USERID, FILE, ISN, ID, etc. followed by each field listed above (AA, NW, CCC, CC1-5, and CG).

SHOWing long lists of fields is not especially effective for identifying an occasionally changed field. For this purpose, AUDITRE features the 'AUDIT' statement. For example:

```
REPORT  TYPE=DETAIL,HEADING='AUDIT FILE 3'
INCLUDE FNR=3
DISPLAY DATE,TIME,SEQ,RESTART-USERID,USERIDX,FNR,ISN
AUDIT   AA*,NW,CCC,CC1-5,CG,FNR=3
```

The output generated is:

```
AUDIT FILE 3                  AUDITRE 5.2 MON 14-01-31    17:45:36

YY-DDD   HH-MM-SS    SEQ     RES-UID   USER-IDX    FNR     ISN

 ********** RECORD UPDATED **********
14-031   15:03:17     30     TREE2     31EF9E2B      3       4

*   AA= HEX 000000000186F4  DEC 100,084                          PERSONNEL-NUMBER
 B: NW=             1111                                        NET-WORTH
 A: NW=             2222                                        NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:17     30     TREE2     31EF9E2B      3       4

*   AA= HEX 000000000186F5  DEC 100,085                          PERSONNEL-NUMBER
 B: NW=             3333                                        NET-WORTH
 A: NW=             4444                                        NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20     34     TREE2     31EF9E2B      3       6

*   AA= HEX 000000000186F6  DEC 100,086                          PERSONNEL-NUMBER
 B: NW=             5555                                        NET-WORTH
 A: NW=             6666                                        NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20     36     TREE2     31EF9E2B      3       7

*   AA= HEX 000000000186F7  DEC 100,087                          PERSONNEL-NUMBER
 B: NW=             7777                                        NET-WORTH
 A: NW=             8888                                        NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE

 NUMBER OF INPUT RECORDS:                          36
 INPUT RECORDS INCLUDED:                            8
 AUDIT DELETES:                                     0
 AUDIT UPDATES:                                     4
 AUDIT ADDS:                                        0
```

Note that the listed fields, key fields, and non-key fields on the AUDIT statement are examined for changes. If key fields are changed, their before and after values are printed. If non-key fields are changed, their before and after values (note the B: and A: in the previous sample) are printed, and the key fields are also printed. If none of the listed fields are changed, none are printed. Key fields do not have to be ADABAS descriptors.

In this example, it was useful to see AA (PERSONNEL-NUMBER) for those persons whose NW (NET-WORTH) changed.

For record Adds and Deletes, AUDITRE provides options of printing all listed fields or only key fields.

It is possible to run multiple logs, generate multiple reports, 'SHOW' specific files' updates, 'AUDIT' other files' updates, print reports, and/or output flat-file records for later processing. Following is a report on two files together.

```
REPORT  TYPE=DETAIL,HEADING='AUDIT FILES 1 AND 3'
INCLUDE FNR=(1,3)
DISPLAY DATE,TIME,SEQ,UIDX,FNR,ISN
AUDIT   AA*,NW,CG,CCC,CC1-5,BK,CR,FNR=3
AUDIT   AA*,BA*,BB,BC,CB,LA,FNR=1
```

This will produce:

```
AUDIT FILES 1 AND 3         AUDITRE 5.2 MON 14-01-31 17:45:36

YY-DDD  HH-MM-SS    SEQ      USER-IDX        FNR     ISN

 ********** RECORD UPDATED **********
14-031  15:03:45     14      31EF9E2B          1      10

*   AA= HEX 000000000186F4  DEC 100,084                                  PERSONNEL-NUMBER
*   BA=KING                                                              NAME
 B: CB=45                                                                AGE
 A: CB=39                                                                AGE
 B: LA=PERFORMANCE MONITORING                                            HOBBY
 A: LA=AUDIT TRAILING                                                    HOBBY

 ********** RECORD UPDATED **********
14-031  15:03:49     16      31EF9E2B          1      11

*   AA= HEX 000000000186F5  DEC 100,085                                  PERSONNEL-NUMBER
*   BA=SPEISER                                                           NAME
 B: CB=45                                                                AGE
 A: CB=39                                                                AGE
 B: LA=PERFORMANCE MONITORING                                            HOBBY
 A: LA=AUDIT TRAILING                                                    HOBBY

 ********** RECORD UPDATED **********
14-031  15:03:49   18        31EF9E2B          1      12

*   AA= HEX 000000000186F7  DEC 100,087                                  PERSONNEL-NUMBER
*   BA=TANIMOTO                                                          NAME
 B: CB=45                                                                AGE
 A: CB=39                                                                AGE
 B: LA=PERFORMANCE MONITORING                                            HOBBY
 A: LA=AUDIT TRAILING                                                    HOBBY
```

```
 ********** RECORD UPDATED **********
14-031  15:03:17   30        31EF9E2B          3       4

*   AA= HEX 000000000186F4  DEC 100,084                                  PERSONNEL-NUMBER
 B: NW=             8888                                                 NET-WORTH
 A: NW=             9999                                                 NET-WORTH
 B: CG=GROVE CITY                                                        COLLEGE
 A: CG=SLIPPERY ROCK                                                     COLLEGE

 ********** RECORD UPDATED **********
14-031  15:03:19   32        31EF9E2B          3       5

*   AA= HEX 000000000186F5  DEC 100,085                                  PERSONNEL-NUMBER
 B: NW=             8888                                                 NET-WORTH
 A: NW=             9999                                                 NET-WORTH
 B: CG=GROVE CITY                                                        COLLEGE
 A: CG=SLIPPERY ROCK                                                     COLLEGE

 ********** RECORD UPDATED **********
14-031  15:03:20   34        31EF9E2B          3       6

*   AA= HEX 000000000186F6  DEC 100,086                                  PERSONNEL-NUMBER
 B: NW=             8888                                                 NET-WORTH
 A: NW=             9999                                                 NET-WORTH
 B: CG=GROVE CITY                                                        COLLEGE
 A: CG=SLIPPERY ROCK                                                     COLLEGE

 ********** RECORD UPDATED **********
14-031  15:03:20   36        31EF9E2B          3       7

*   AA= HEX 000000000186F7  DEC 100,087                                  PERSONNEL-NUMBER
 B: NW=             8888                                                 NET-WORTH
 A: NW=             9999                                                 NET-WORTH
 B: CG=GROVE CITY                                                        COLLEGE
 A: CG=SLIPPERY ROCK                                                     COLLEGE

 NUMBER OF INPUT RECORDS:     36
 INPUT RECORDS INCLUDED:      14
 AUDIT DELETES:        0
 AUDIT UPDATES:        7
 AUDIT ADDS:           0
```

For two (or more) files that are logically related, the example above shows how intermixed file updates can be reported. In this example, the PERSONNEL (1) and FINANCE (3) files are related, both having a PERSONNEL-NUMBER (AA).

AUDITRE can also be used to select all the records from one or more files for output to a 'protection log like' file (dataset). Simply follow the above parameters with:

```
  OUTPUT COMPRESS=YES
```

This will result in a new PLOG-like file to contain only file 1 and file 3 compressed PLOG images. An installation may normally generate multiple PLOG reels per day. These can be run through AUDITRE and reduced down to one or two reels containing only the files from which AUDITRE reports will eventually be produced.

To obtain a summary by file and user, code one or more reports such as:

```
REPORT  TYPE=SUMMARY,
        HEADING='SUMMARY OF PLOG UPDATES BY FILE/USER'
CONTROL FNR,UIDX

REPORT  TYPE=SUMMARY,
        HEADING='SUMMARY OF PLOG UPDATES BY HOUR/FILE'
CONTROL HOUR,FNR
```

The output will be:

SUMMARY OF PLOG UPDATES BY FILE/USER

```
    FNR        USER-IDX          COUNT                 %
      1        31EF9E2B              6              16.7
      1        ******                6              16.7
      3        31EF9E2B              8              22.2
      3        ******                8              22.2
      8        31EF9E2B             21              58.3
      8        ******               21              58.3
   4319        FFFFFF00              1               2.8
   4319        ******                1               2.8

    ***        ******               36             100.0

NUMBER OF INPUT RECORDS:              36
INPUT RECORDS INCLUDED:               36


SUMMARY OF PLOG UPDATES BY HOUR/FILE

       HR          FNR          COUNT               %
       14         4319              1               2.8
       14           **              1               2.8
       15            1              6              16.7
       15            3              8              22.2
       15            8             21              58.4
       15           **             35              97.2

       **           **             36             100.0

NUMBER OF INPUT RECORDS:              36
INPUT RECORDS INCLUDED:               36
```

These summary reports indicate updates were made to files 1, 3, and 8. To see which NATURAL programs and modules were updated, code the following:

```
REPORT TYPE=DETAIL,
       HEADING='NAT PROGRAMS, MODULES UPDATED'
INCLUDE FNR=8
DISPLAY WEEKDAY,TIME
AUDIT   LJ*,LK1*,LL*,FNR=8
```

For a SAVE/CAT/STOW, NATURAL completely deletes the old program/module, and then adds the entire new program/module to the NATURAL System File. A few records showing this update process are displayed below:

```
NAT PROGRAMS, MODULES UPDATED     AUDITRE 5.2 MON 14-01-31 17:45:36

DAY      HH-MM-SS


 ********** RECORD DELETED **********
TUE      15:01:36

A: LJ=                                                          SOURCE-PROG-ID
A: LK  1=                                                       SOURCE-CODE
A: LL=PERSONL UPDPERS                                           OBJECT-PROG-ID

 ********** RECORD ADDED **********
TUE      15:01:37

A: LJ=                                                          SOURCE-PROG-ID
A: LK  1=                                                       SOURCE-CODE
A: LL=PERSONL UPDPERS                                           OBJECT-PROG-ID

 ********** RECORD DELETED **********
TUE      15:01:37

A: LJ=PERSONL UPDPERS                                           SOURCE-PROG-ID
A: LK1=*TSIT2 15:01:3807-11-24V4.1NAT                           SOURCE-CODE
A: LL=                                                          OBJECT-PROG-ID

 ********** RECORD ADDED *******
TUE       15:01:37

A: LJ=PERSONL UPDPERS                                           SOURCE-PROG-ID
A: LK1=*TSIT2 15:01:3807-11-24V4.1NAT                           SOURCE-CODE
A: LL=                                                          OBJECT-PROG-ID

 ********** RECORD DELETED *******
TUE 15:02:35

A: LJ=                                                          SOURCE-PROG-ID
A: LK  1=                                                       SOURCE-CODE
A: LL=PERSONL FINANC                                            OBJECT-PROG-ID

 ********** RECORD ADDED **********
TUE 15:02:49

A: LJ=                                                          SOURCE-PROG-ID
A: LK  1=                                                       SOURCE-CODE
A: LL=PERSONL FINANC                                            OBJECT-PROG-ID

 ********** RECORD DELETED **********
TUE 15:03:06

A: LJ=PERSONL FINANC                                            SOURCE-PROG-ID
A: LK 1=*TSIT2 13:48:5507-11-24V4.1NAT                          SOURCE-CODE
A: LL=                                                          OBJECT-PROG-ID
```

```
********** RECORD ADDED *******
TUE 15:03:06

A: LJ=PERSONL FINANC                                              SOURCE-PROG-ID
A: LK 1=*TSIT2 15:03:0207-11-24V4.1NAT                            SOURCE-CODE
A: LL=                                                            OBJECT-PROG-ID


 NUMBER OF INPUT RECORDS:                  36
 INPUT RECORDS INCLUDED:                   21
 AUDIT DELETES:                            10
 AUDIT UPDATES:                             0
 AUDIT ADDS:                               11
```

Examination of the report above reveals that the source (LJ) and the object (LL) programs are being manipulated. (LK1 is the 1st line of the source.)

To see all of the fields updated on file 3, a long list of fields could be stated for a SHOW report. It would be better to use an AUDIT report, which will do the comparison of Before and After Images of each field and identify the changes. It may be best to use the 'AUDIT ALL' feature. This feature is provided so that all the individual fields do not have to be listed. The power of the 'AUDIT ALL' is in the reports it produces. Consider the following parameters:

```
REPORT  TYPE=DETAIL,HEADING='UPDATES ON FILE 3'
INCLUDE FNR=3
DISPLAY DATE,TIME,USERIDX,FNR,ISN
AUDIT   ALL,FNR=3
```

The following report will be generated:

```
UPDATES ON FILE 3           AUDITRE 5.2 MON 14-01-31 17:45:36

YY-DDD   HH-MM-SS   USER-IDX  FNR   ISN

 ********** RECORD UPDATED **********
14-031   15:03:17   31EF9E2B    3     4

B: NW=                          8888                                NET-WORTH
A: NW=                          9999                                NET-WORTH
B: CG=GROVE CITY                                                    COLLEGE
A: CG=SLIPPERY ROCK                                                 COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:19   31EF9E2B    3     5

B: NW=                          8888                                NET-WORTH
A: NW=                          9999                                NET-WORTH
B: CG=GROVE CITY                                                    COLLEGE
A: CG=SLIPPERY ROCK                                                 COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20   31EF9E2B    3     6

B: NW=                          8888                                NET-WORTH
A: NW=                          9999                                NET-WORTH
B: CG=GROVE CITY                                                    COLLEGE
A: CG=SLIPPERY ROCK                                                 COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:19   31EF9E2B    3     7

B: NW=                          8888                                NET-WORTH
A: NW=                          9999                                NET-WORTH
B: CG=GROVE CITY                                                    COLLEGE
A: CG=SLIPPERY ROCK                                                 COLLEGE


 NUMBER OF INPUT RECORDS:                  36
 INPUT RECORDS INCLUDED:                    8
 AUDIT DELETES:                             0
 AUDIT UPDATES:                             4
 AUDIT ADDS:                                0
```

AUDITRE will consider the report on the previous page as if it had been stated to audit for fields AA, MCC (the count of occurrences of MC), CC1-10 (occurrences 1-10 of CC), CL1-10, CB1-10, OC, NW, CR, IPC, IC1C, IC1#1-10 (MU occurrences 1-10 of occurrence 1 of periodic field IC), PA1C, PA1#1-10, (IC and PA fields will be repeated for PE occurrences 2-10) CG, VCC, OV1-10, IV, SV, and BK. Ten occurrences of MU and PE fields is the default. In other words, AUDITRE decompresses every field and lists every field changed on file 3, including applicable occurrences. AUDITRE gets these occurrence ranges based upon ADACMP file definition occurrence values. Only the changed fields will be printed on as small a report as possible.

If the report indicates some record's NET-WORTH rose drastically, management might want to know who this belongs to (i.e., via the key field PERSONNEL-NUMBER). One could code:

```
  AUDIT AA*,ALL,FNR=3
```

This means the value of AA will be printed for any record that changed. For the personnel file one could code:

```
  AUDIT BA*,BB*,AA*,ALL,FNR=1
```

This means the value of LAST-NAME, FIRST-NAME, and PERSONNEL-NUMBER will be printed for any updated records on file 1.

To generate an audit of the PERSONNEL and FINANCE files, code:

```
REPORT  TYPE=DETAIL,HEADING='PERS, FINANCE AUDIT'
INCLUDE FNR=(1,3)
DISPLAY DATE,TIME,RESTART-USERID,FNR
AUDIT   BA*,BB*,AA*,ALL,FNR=1
AUDIT   AA*,ALL,FNR=3
```

The printout will be:

```
PERS, FINANCE AUDIT    AUDITRE 5.2     MON 14-01-31   17:45:36

YY-DDD   HH-MM-SS   RES-UID   FNR

 ********** RECORD UPDATED **********
14-031   15:01:45   TREE2   1

*   BA=KING                                                    NAME
*   BB=EDDIE                                                   FIRST-NAME
*   AA= HEX 000000000186F4  DEC 100,084                        PERSONNEL-NUMBER
 B: CB=45                                                      AGE
 A: CB=39                                                      AGE
 B: LA=PERFORMANCE MONITORING                                  HOBBY
 A: LA=AUDIT TRAILING                                          HOBBY

 ********** RECORD UPDATED **********
14-031   15:01:49   TREE2   1

*   BA=SPEISER                                                 NAME
*   BB=PHYLLIS                                                 FIRST-NAME
*   AA= HEX 000000000186F5  DEC 100,085                        PERSONNEL-NUMBER
 B: CB=45                                                      AGE
 A: CB=39                                                      AGE
 B: LA=PERFORMANCE MONITORING                                  HOBBY
 A: LA=AUDIT TRAILING                                          HOBBY
```

```
 ********** RECORD UPDATED **********
14-031   15:01:49   TREE2   1

*   BA=TANIMOTO                                                  NAME
*   BB=DAVID                                                     FIRST-NAME
*   AA= HEX 000000000186F7  DEC 100,087                          PERSONNEL-NUMBER
 B: CB=45                                                        AGE
 A: CB=39                                                        AGE
 B: LA=PERFORMANCE MONITORING                                    HOBBY
 A: LA=AUDIT TRAILING                                            HOBBY

 ********** RECORD UPDATED **********
14-031   15:03:17   TREE2   3

*   AA= HEX 000000000186F4  DEC 100,084                          PERSONNEL-NUMBER
 B: NW=        8888                                              NET-WORTH
 A: NW=        9999                                              NET-WORTH
 B: CG=GROVE CITY                                                COLLEGE
 A: CG=SLIPPERY ROCK                                             COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20   TREE2   3

*   AA= HEX 000000000186F5  DEC 100,085                          PERSONNEL-NUMBER
 B: NW=        8888                                              NET-WORTH
 A: NW=        9999                                              NET-WORTH
 B: CG=GROVE CITY                                                COLLEGE
 A: CG=SLIPPERY ROCK                                             COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20   TREE2   3

*   AA= HEX 000000000186F6  DEC 100,086                          PERSONNEL-NUMBER
 B: NW=        8888                                              NET-WORTH
 A: NW=        9999                                              NET-WORTH
 B: CG=GROVE CITY                                                COLLEGE
 A: CG=SLIPPERY ROCK                                             COLLEGE

 ********** RECORD UPDATED **********
14-031   15:03:20   TREE2   3

*   AA= HEX 000000000186F7  DEC 100,087                          PERSONNEL-NUMBER
 B: NW=        8888                                              NET-WORTH
 A: NW=        9999                                              NET-WORTH
 B: CG=GROVE CITY                                                COLLEGE
 A: CG=SLIPPERY ROCK                                             COLLEGE


 NUMBER OF INPUT RECORDS:                  36
 INPUT RECORDS INCLUDED:                   14
 AUDIT DELETES:                             0
 AUDIT UPDATES:                             7
 AUDIT ADDS:                                0
```

Many fields can be placed before and after the 'ALL'. For example:

```
AUDIT   AA*,CB1-2*,ALL,AA,NW,ALL,AA*,FNR=3
```

This will identify all changed fields twice, with AA, CB1-2, and NW (if changed) listed multiple times. One record of the AUDIT would be shown as:

```
FNR   ISN     DATE      TIME     RES-UID
  3     7    07-365   15:03:20   TREE2
 *** RECORD UPDATED

*   AA= HEX 000000000186F7  DEC 100,087                          PERSONNEL-NUMBER
*   CB  1=0080                                                   CURRENT-BALANCE
*   CB  2=1200                                                   CURRENT-BALANCE
 B: NW=            8888                                         NET-WORTH
 A: NW=            9999                                         NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE
 B: NW=            8888                                         NET-WORTH
 A: NW=            9999                                         NET-WORTH
 B: NW=            8888                                         NET-WORTH
 A: NW=            9999                                         NET-WORTH
 B: CG=GROVE CITY                                               COLLEGE
 A: CG=SLIPPERY ROCK                                            COLLEGE
*   AA= HEX 00000000000186F7  DEC 100,087                        PERSONNEL-NUMBER
```

This example, however illogical, illustrates the power, flexibility, and thoroughness of AUDITRE.

At the end of each report, totals of Updates, Adds, and Deletes are shown for each file and for each field within each file as listed in the AUDIT statement. An example for the earlier report entitled 'AUDIT FILES 1 AND 3' follows:

```
FILE:   3         DELETES:        0        UPDATES:        4       ADDS:        0

FIELD  LONG NAME           OCC  FROM  OCC TO  OCC FROM  OCC TO    DELETES UPDATES    ADDS
AA     PERSONNEL-NUMBER                                          0       0       0
CC     CREDIT-CARD                  1                            4       0       0  0
OC     OIL-CREDIT              1    10                           0       0       0
NW     NET-WORTH                                                 0       4       0
CR     CREDIT-RATING                                             0       0       0
CG     COLLEGE                                                           0       4  0
BK     BANK                                                      0       0       0


FILE:   1         DELETES:        0       UPDATES:         3         ADDS:        0

FIELD  LONG NAME           OCC  FROM  OCC  TO  OCC FROM  OCC TO   DELETES UPDATES    ADDS
AA     PERSONNEL-NUMBER                                          0       0       0
BA     LAST-NAME                                                 0       0       0
BB     FIRST-NAME                                                0       0       0
BC     INITIAL                                                   0       0       0
CB     AGE                                                       0       3       0
LA     HOBBY                                                             0       3  0
```

This summary report may highlight unapproved changes to the file, such as changes to NET-WORTH when there should not be any or more SALARY updates than expected, etc.

APPENDIX D

DATE PROCESSING

AUDITRE obtains the STCK time from the PLOG. From this STCK value, AUDITRE can create year, month, day, hour, minute, and second values, as well as other derived values, such as month-name (January, etc.). Four fields were added to the "derived fields" in AUDITRE. These are:

- DATE4
- YYYYMMDD
- DATE4TIME
- YEAR4

These four fields can be considered "extensions" of the standard four derived fields: DATE, YYMMDD, DATETIME, and YEAR. The four fields represent the year portion as four digits rather than two. These fields can be used in all appropriate AUDITRE batch processing parameter statements, such as:

- DISPLAY FNR,CMD,NATPROG,**YEAR4**,MONTH,WEEKDAY,TIME,...
- INCLUDE FNR=(1234-1238),**YYYYMMDD**=20111219,...
- CONTROL **DATE4**,HR,JOB,...
- MAXIMUM **DATE4TIME**,...

It will be necessary for the user to allow for the additional two digits on reports, in output datasets produced by AUDITRE, in any pre-processing to produce AUDITRE parameters, or in any post processing of AUDITRE outputs. The exact layout of these fields is described in **Section III Log Analysis Parameter Statements**.

The 2-digit date formats were retained. However, beyond the year 2000, use of these old style date formats may cause problems for the user. For example, a batch AUDITRE run with an INCLUDE based on the two dates (99 to 00) would look illogical to AUDITRE, and no records would be included. Using the new style date formats would cause no problem with an INCLUDE from 1999 to 2000.

The AUDITRE INPUT statement allows for STOPDATE/STOPTIME, and STARTDATE/STARTTIME combinations. Full explanation of how to use these combinations is presented earlier in this manual. As an option a four-digit date can be specified. It is possible to state STOPDATE4=20120131 or STOPDATE=120131, for example. These are both interpreted as stopping PLOG processing on January 31 of 2012..

Appendix B Output Dataset Formats describes the output formats for the various types of output datasets produced by AUDITRE batch processing. In an output dataset related to a summary report, the 80-byte header contains three dates: AUDITRE report date; starting date; and ending date of the processed log data. There are two ways to have these dates output. The first way is the 2-digit style (e.g., 12-01-31). This style is created when the OUTPUT statement contains the parameter YEARFMT=2 (this is the default). The second way is the 4-digit style, e.g., 20120131. This style is created when the OUTPUT statement contains the parameter YEARFMT=4.

In an output dataset for PLOG data with the COMPRESS=NO option, there is an 80-byte header, with two dates: AUDITRE report date; and date of the PLOG-reported data modification. There are two ways to have these dates output. The first way is the 2-digit style (e.g., 12-01-31). This style is created when the OUTPUT statement contains the parameter YEARFMT=2 (this is the default). The second way is the 4-digit style (e.g., 20120131). This style is created when the OUTPUT statement contains the parameter YEARFMT=4.

With the date-related enhancements described above, users can specify the format of dates in AUDITRE batch reports and outputs.

INDEX