Advantage™ CA-PanAudit® Plus

Macro Reference Guide

3.0

Computer Associates®

Contents

Chapter 1: Introduction

Chapter 2: Using Routines

Chapter 3: Coding Guidelines

Chapter 4: Generalized/Statistical Routines A-C

Chapter 5: Generalized/Statistical Routines D-E

Chapter 6: Generalized/Statistical Routines F-N

Chapter 7: Generalized/Statistical Routines O-R

Chapter 8: Generalized/Statistical Routines S-Z

Chapter 9: Basic SMF Reporting Facility

Chapter 10: Advanced SMF Reporting Facility (JIF)

Chapter 11: Graphing Facility

Chapter 12: Advanced Techniques

Appendix A: Keywords

Appendix B: Sample Applications

Index

Chapter 1

Introduction

Following is a partial list of the technical aspects of CA-PanAudit Plus keyed to their presentation in this guide.

The CA-PanAudit Plus *Macro Reference Guide* is both an audit guide and a training guide. The modular format lets you select the functions most appropriate to your audit objectives. Guidelines that can be expanded or modified to suit your environment are provided for performing a thorough, computer-based audit.

A Tool for the Auditor

Computer-based information systems have increased significantly in the past few years, and all indicators predict further increases in their use. Most of the primary business activities will eventually be computerized.

The computer age has not changed the basic objectives of financial or operational auditing, but three major areas are affected by computer technology:

- Audit independence
- Methods of obtaining evidence
- Evaluation of internal controls

Because many accounting functions are now automated, new techniques are needed to establish the desired levels of control, and the auditor must develop procedures to take advantage of data processing technology.

CA-PanAudit Plus

CA-PanAudit Plus is a computer-based auditing system using audit-specific statistical routines to test and report on the information stored in computer files.

Input

A CA-PanAudit Plus job evaluates the data in a computer file and enables you to access and process it quickly and easily. Each file consists of a set of related records (a personnel file may have one record per employee). Typically, a computer file is constructed so that related pieces of information (for example, names or employee numbers) are located in the same relative position in each record. These locations within records are called fields. A CA-PanAudit Plus program finds the field you want, takes the information you request, and processes it in the manner you specify.

With CA-PanAudit Plus, you can retrieve any kind of record from any kind of file structure, including complex database structures such as IMS, DLI, and IDMS. You can write your own CA-PanAudit Plus programs for every audit task, ensuring you the independence necessary for a valid audit.

Output

CA-Easytrieve Plus provides four types of output:

- Printed output of any kind (reports, letters, forms, mailing labels, and so on)
- Punched card output
- File output
- Summary reports and summary output files

Host Language

CA-PanAudit Plus is a collection of routines written in CA-Easytrieve Plus . These routines invoke CA-Easytrieve Plus macros and issue calls to routines written in other languages. Therefore, the execution of a CA-PanAudit Plus routine is merely the execution of a CA-Easytrieve Plus job or jobs. The basic structures that apply to CA-PanAudit Plus are the exact same structures that exist in CA-Easytrieve Plus .

In all CA-PanAudit Plus documentation, the name CA-Easytrieve Plus refers to the language that CA-PanAudit Plus uses to accomplish its tasks. It indicates that the document is referring to technical aspects of the host language. The name CA-PanAudit Plus refers to the overall characteristics of the product and to the individual routines written in CA-Easytrieve Plus .

To use CA-PanAudit Plus effectively you need not know CA-Easytrieve Plus . The use of CA-PanAudit Plus routines can be taught to personnel having no formal data processing background.

Capabilities

CA-PanAudit Plus, the Computer Auditing System, provides you with the tools, techniques, and training for performing computer-based audits.

By computerizing statistical auditing procedures with CA-PanAudit Plus, you enhance your ability to:

- Perform more comprehensive audits
- Process data on large populations economically
- Measure the reliability of the estimate obtained from a sample
- Provide a more objective result from the sampling

Ease of Use

CA-PanAudit Plus uses English-like statements to execute routines specific to auditing applications. It is designed for both EDP auditors and financial auditors. The basic features that make CA-PanAudit Plus easy to use are:

- English-language format
- Automatically formatted reports
- Self-documenting audit trails
- Boolean logic (IF, AND, OR) and relational operators (greater than, equal to, and so on)
- FILE statements
- Standard numeric computation

Related Publications

The following documents are available for use with CA-PanAudit Plus:

CA-Easytrieve

CA-Easytrieve*Programmer Guide.* Detailed information and advanced coding techniques in CA-Easytrieve.

CA-Easytrieve*Introduction to the Language.* An introduction to CA-Easytrieve, the CA-PanAudit Plus host language. Use this guide before you begin writing your own programs.

CA-Easytrieve*Language Reference Guide*. This guide is your source for complete system details. It contains descriptions of all product features and functions and summaries of each CA-Easytrieveversion.

CA-Easytrieve Plus *Extended Reporting Facility*. Describes support of extended reporting for Impact Dot, Ink Jet, and Electro Photographic printers.

CA-Easytrieve Plus *Interface Options Guides*. These are short guides available for users of various system options. There are guides for IMS/DLI processing, CA-IDMS, and IDD processing, TOTAL processing, SQL processing, CA-Datacom/DB processing, SUPRA processing, and other CA-Easytrieve Plus options.

Command Notation

The following conventions are used throughout this guide:

bold	**Bold** text in program code is used to highlight an example of the use of a statement. Text that you must enter into an input field exactly as shown is in **bold**.
lowercase	A variable that you must supply in a statement syntax appears in lowercase.
{braces}	Required choice of one of these entries.
[brackets]	Optional entry or choice of one of these entries.
| (OR bar)	Choice of one of these entries.
(parentheses)	Multiple parameters must be enclosed in parentheses.
...	Ellipses indicate that you can code the immediately preceding parameters multiple times.
CAPS	All capital letters indicate a keyword, a name or a field used in a program example.

Chapter

2

Using Routines

This chapter describes how to use CA-PanAudit Plus generalized and statistical routines.

Generalized Routines

In addition to statistical analyses, an audit involves various types of data verification and testing. Many of these formerly time-consuming tasks are now automated. CA-PanAudit Plus generalized routines are designed to perform a wide array of the tasks common to general data analysis—from numerical analysis to test data generation. The routines fall into six categories:

- Integrity tests
- Date and time routines
- Numeric routines
- File comparison
- Test data generation
- Miscellaneous routines

These generalized routines can often be used with other generalized Routines O-R in preparation for statistical routines.

A brief description of the generalized routines, by category, is given on the following pages. A detailed description of each routine, listed in alphabetical order, is given in later chapters.

Integrity Tests

In conducting an audit, one of your major functions is to verify the validity of data in computer files. CA-PanAudit Plus integrity tests assist you in performing various types of compliance tests.

The integrity test routines are both inline and stand-alone CA-PanAudit Plus routines. For a description of how to use inline and stand-alone routines, see the chapter "Coding Guidelines."

The integrity test routines are:

DATEVAL	Tests if the content of a specified date field is valid for a given date format. Sets a flag that reports the results of the test.
DUPTEST	Tests a field to determine if the content appears in more than one record and reports on the results.
FLDVALR	Tests a field for a range of values and sets a flag to report on the result of the test. Reports on the results and, optionally, writes to an output file.
FLDVALT	Tests a field for values found in a table and sets a flag to report on the result of the test. Reports on the results and, optionally, writes to an output file.
FLDVALV	Tests a field for a specific value and sets a flag to report on the result of the test. Reports on the results and, optionally, writes to an output file.
GAPCHCK	Tests a numeric field to determine if records in a continuous sequence are missing from the file. Produces a report of the missing numbers.
MULTDUP	Compares up to 50 fields to determine if duplicate records exist. Optionally, writes duplicate records to an output file.
NUMTEST	Tests if the content of a specified field is numeric. Prints hexadecimal representations of non-numeric data.

Date and Time Routines

You can easily perform date and time conversions and computations by using one of the CA-PanAudit Plus date and time routines. Some routines require you to define a field to hold the result of the computation or require you to interrogate a previously defined field to test the results of a computation.

All date and time routines are inline CA-PanAudit Plus routines. For a detailed description of how to use inline routines, see the chapter "Advanced Techniques."

Date and time routines become useful when used with other CA-PanAudit Plus routines. For example, you may want to produce a distribution analysis of clients who took over 60 days to pay an invoice. This is accomplished using the DAYSCALC routine and one of the distribution analysis routines.

The date and time routines are:

DATECALC	Adds or subtracts a given number of days from the date in a specified field and writes the resulting date to a second field.
DATECONV	Reformats a date from a specified format to another format (for example, Gregorian to Julian, Julian to Gregorian).
DAYSAGO	Calculates the number of elapsed days between the current date and the date in a specified field. The result is compared to a specified value, and an indicator is set if a specified criterion is met (for example, EQ, GR, GQ, LS, LQ, or NQ).
DAYSCALC	Calculates the number of elapsed days between two dates. The result is written to a specified field.
GETDATE	Gets the current date from the system and places it in a specified field.
TIMECONV	Converts time from hundredths of a second to hours, minutes, seconds, and hundredths of seconds.
WEEKDAY	Calculates the day of the week for a given date and positions it in a field for further use.

Numeric Routines

The numeric CA-PanAudit Plus routines perform a variety of numerical calculations. All numeric routines are inline CA-PanAudit Plus routines. For a detailed description of how to use inline routines, see the chapter "Advanced Techniques."

The numeric routines are:

DIVIDE	Calculates the quotient and remainder of the division of two integers.
EXPO	Calculates the exponentiation of numeric values.
RANDOM	Produces a series of random numbers from 1 to 15 digits in length.
RANDSPAN	Produces a series of random numbers in a specified range.
SQRT	Calculates the square root of a number to an accuracy of two decimal places.
STDDEV	Calculates the standard deviation of a set of variables for designated intervals in the file and for the entire file.

File Comparison

The file comparison routines are used to compare numeric data, source code, or any other form of data. They are stand-alone CA-PanAudit Plus routines. For a detailed description of how to use stand-alone routines, see the chapter "Coding Guidelines."

The file comparison routines are:

FILECOMP — Compares specified areas of records in two files and produces a report of mismatched records. Contains a method for realigning the files when a mismatch occurs. Use the facility to compare any type of data.

SRCECOMP — Compares two versions of a source program and prints a report of all added, deleted, and changed statements.

File Compare Facility

Use the file compare facility to ensure the integrity of two files by comparing selected fields. You can compare source code, object modules, and data files.

You can make comparisons on a strict record-by-record basis or realign files using designated keys (see following explanation). When a mismatch occurs between a record pair in the primary and secondary files, both records are printed in hexadecimal, along with their character representation.

File Compare Keys

A key is a portion of the record for which a match is searched so the files can be repositioned after an unequal record pair causes a mismatch. Any field in the input record can be used as a key.

You can designate from one to six keys. Keys, if used, must be specified for both the primary and secondary files. They must be defined in the library section of each file and are specified in the PRIKEYS and SECKEYS parameters of the FILECOMP routine. For additional information, see FILECOMP in the chapter "Generalized/Statistical Routines F-N."

When choosing keys, use alphanumeric or numeric fields where the content is in ascending sequence, for example, an alphabetic name field in a data file or the statement number field in source code.

You must specify keys in pairs (for example, for each key specified in PRIKEYS, you must specify an associated key in SECKEYS). If multiple keys are used, all key fields must match before the files are realigned.

Specification of keys is optional. However, if no key is specified, automatic repositioning cannot be performed. The files are compared on a record-by-record basis. This means if a mismatch occurs because of an unequal record pair, the remainder of the file is interpreted as being mismatched, unless records become realigned by chance.

Examples

The two examples that follow show how files are compared on a record-by-record basis (no keys specified) and when keys are specified for the comparison.

File Compare Process Without Keys

```
Primary       Secondary
 File           File
| 1  | ------- | 1  |
| 2  | ------- | 2  |
| 3  | - - - - | 4  |  > 3,4
| 5  | - - - - | 6  |  > 5,6
| 6  | - - - - | 7  |  > 6,7   These 5
| 7  | - - - - | 8  |  > 7,8   record
| 8  | - - - - | 9  |  > 8,9   pairs are
| 10 | ------- | 10 |          printed
| 14 | ------- | 14 |
| 15 | ------- | 15 |
| 16 | ------- | 16 |
```

This simplified example depicts the file compare process when no key is specified. Comparison is made on a record-by-record basis, and no automatic repositioning takes place. In this example, record pair (3,4) causes a mismatch for the next five record pairs. The files become realigned with record pair (10,10).

File Compare Process With Keys

```
 Primary        Secondary
  File            File
 KEY1A            KEY2A

|  1  |  ——————  |  1  |
|  2  |  ——————  |  2  |
|  3  |  - - - - |  4  |  > 3,4
                         > 5,4
|  5  |  - - - - |  6  |  > 5,6
                                  These 4
|  6  |          |  7  |          record
                                  pairs are
|  7  |          |  8  |          printed
|  8  |          |  9  |  > 10,9
| 10  |  ——————  | 10  |
| 14  |  ——————  | 14  |
| 15  |  ——————  | 15  |
| 16  |  ——————  | 16  |
```

In this example, one pair of keys (KEY1A, KEY2A) is defined. The first mismatch occurs between record pair (3,4). When an attempt is made to realign the files, record pairs (5,4) and (5,6) are also mismatched. Alignment is achieved again by matching the key fields in record pair (6,6). Another mismatch occurs between (10,9), and the files are aligned again at (10,10).

For simplicity, this example illustrates the remainder of the record matches when the key fields match. However, even when keys match, if the specified comparison fields in the two records are not exactly identical, the record pair is diagnosed as mismatched, and printed.

File Definitions Required

You must code two FILE statements before invoking FILECOMP:

- One for the primary file
- One for the secondary file

You must also define the optional key fields at this time.

The following is an example of FILE statements for the FILECOMP routine:

```
FILE PRIMARY
  PRIM-ACCT-NUM     22   4  N
  PRIM-NAME         76  20  A
  ...
FILE SECONDARY
  SEC-ACCT-NUM      22   4  N
  SEC-NAME          76  20  A
  ...
```

In this example, two pairs of keys (PRIM-ACCT-NUM, SEC-ACCT-NUM and PRIM-NAME, SEC-NAME) are defined. The first pair are numeric and begin in position 22 for a length of 4; the second pair are alphanumeric and begin in position 76 for a length of 20.

Test Data Generation

CA-PanAudit Plus gives you the ability to generate specified data (alphanumeric, invalid, dates, and numeric) in user-defined formats that can be used for audit tests in place of actual production data.

Each routine is an inline CA-PanAudit Plus routine. However, to successfully generate test data, you use the routines as a system—starting with one invocation of FILEGEN followed by multiple invocations of ALPHAGEN, BADGEN, DATEGEN, or NUMGEN.

Test Data Generator Facility

The test data generator facility creates an output file and generates data in a user-defined format. Files generated with this facility can be used in lieu of existing files that may contain sensitive data. You can generate test files with characteristics identical to production files. Use these files to test Routines O-R as an aid in learning how to use the CA-PanAudit Plus system.

File Generation

The test data generator consists of the FILEGEN routine and four data generation routines:

- ALPHAGEN
- BADGEN
- DATEGEN
- NUMGEN

The FILEGEN routine establishes the output file name, the number of records, the seed for the random number generator, and whether a hexadecimal listing of output records is to be created. After the output file is defined with FILEGEN, use the data generation routines to create data in specified fields in the output records.

Data Generation

Routines to generate data are invoked after the FILEGEN routine. You can invoke each of these routines any number of times and in any combination, to generate data for fields in a file. During execution, each data generation routine writes the data it produces into the field specified in the field name parameter. This process is repeated until data is generated for each record being created. The number parameter of the FILEGEN routine determines the number of test records generated.

Many options exist for the generation of data. Generated data for ALPHAGEN, NUMGEN, and DATEGEN (with the exception noted for DATEGEN) can be:

- Random in a specified range
- Sequenced in increasing or decreasing order
- Incremented by a specified amount
- A series of constants (disallowed in DATEGEN)

A parameter that establishes conditions (such as the minimum or maximum for a range of random values) can either be an actual value or the name of a field containing the value.

The test data generation routines are:

ALPHAGEN	Generates alphanumeric data.
BADGEN	Generates a specified percentage of invalid data.
DATEGEN	Generates dates in any user-defined format.
FILEGEN	Defines the output file to which the data created by the other test data generation routines is written.
NUMGEN	Generates numeric data.

Miscellaneous Routines

CA-PanAudit Plus provides eight routines that do not fit in the preceding categories of generalized routines. The miscellaneous routines are:

ADDRCMP

Reduces addresses to a common key and produces a report of potentially duplicate addresses within the same ZIP code.

AGING

Performs an aging analysis of an input file. AGING calculates the age of a specified field, places it in a category, and produces an aging report. You can customize this report by specifying the base date, the number of intervals, the number of days per interval, whether current items are to be listed, and whether the report is to be a detail or summary report.

ALPHACON

Converts an edited numeric field (an alphanumeric field) into a numeric field.

APR

Calculates the annual percentage rate for a given amount and interest rate.

CBLCNVRT

Converts COBOL data definitions to their CA-Easytrieve Plus equivalent.

CONVAE/CONVEA

Converts ASCII alphanumeric data to EBCDIC and conversely.

DECRYPT/ENCRYPT

The use of cryptographic techniques to protect information from accidental or intentional disclosure is an important aspect of data integrity. Cryptography is most useful when you cannot guarantee protection of files against disclosure, modification, or duplication. Three common situations where you cannot guarantee protection are:

- Copying production data for use in a test environment
- Processing data files at another installation
- Transmitting data over telecommunication lines

A key value controls the decryption and encryption process in CA-PanAudit Plus. The key value initiates a random process to render data unintelligible. Security in the system depends on the secrecy of the key that locks and unlocks the data. The key is a one- to seven-digit value. Treat the key with the same level of protection as the combination to a company's safe. You can change the key value to protect it against unauthorized disclosure. Since you cannot decrypt the data without the key, it can be time consuming and expensive to recreate the data if the key value is lost.

These Routines D-Ecrypt and encrypt data in a file to protect the file against accidental or intentional disclosure.

UNBYTE

Converts a one-byte input field into eight one-byte fields, which correspond to the eight bits of the input byte.

Statistical Routines

The CA-PanAudit Plus statistical routines provide the information you need to design and select a statistically valid sample. Routines are provided in the following three categories:

Distribution Analysis—These routines show the frequency of occurrence of specified values in the input file. These routines are often used to select significant data for further sampling.

Statistical Auditing and Sampling—These routines enable you to sample large populations using various sampling techniques.

Statistical Forecasting—These routines use linear and multiple regression/correlation analysis as the basis for their estimates.

A brief description of the statistical routines, by category, is given on the following pages. A detailed description of each routine, listed in alphabetical order, is given in later chapters.

Distribution Analysis Routines

The distribution analysis routines provide information about the distribution of quantitative and nonquantitative data in a file and produce reports on the frequency of occurrence of selected values in a file.

The distribution analysis routines are stand-alone CA-PanAudit Plus routines and contain an optional technique for graphing percentages on the reports.

These routines become most useful with other CA-PanAudit Plus routines, ensuring that routines specify meaningful parameter values prior to execution. For example, before selecting a cell width in DOLUNIT or a target value in STRATIF, it is helpful to perform an interval analysis (INTERVL).

The distribution analysis routines are:

INTERVL — Reports on the frequency of occurrence of different values in specified quantitative fields. Permits variable interval sizes. Calculates mean and standard deviation.

OCCURS — Reports on the frequency of occurrence of different values in a specified nonquantitative field.

VERSUS — Produces an analysis comparing any quantitative field against any nonquantitative field.

Statistical Auditing and Sampling Routines

The statistical auditing and sampling routines use established statistical methods to process and sample large populations. Use the routines to provide detail reports and to create sample files for auditing procedures. Estimates of reliability and evaluations are provided for most of the routines.

The following table shows the statistical auditing and sampling routines and their uses:

Application	Routine
Sampling for attributes	ATTPCT ATTSAMP
Proportional sampling	CAVEVAL CAVSAMP DOLUNIT SPS
Discovery sampling	DISCPCT DISCSMP
Calculate population sizes	POPCOUNT POPSIZE
Random sampling routines	INTSAMP RANDPCT RANDXCT STOPORGO
Three steps used in regression estimates	REGEVAL REGSAM REGSAMP
Stratified sampling for variables when there is a wide variance in the population	STRATIF STRTEVL
Sampling for variables	VARPCT VARSAMP

Attribute and Variable Sampling

You can use CA-PanAudit Plus routines to sample for attributes or variables.

Sampling for Attributes

Use the attribute sampling routines to estimate the probable frequency that a certain event will occur, given normal distribution in a population. An event can be a particular class of error or any other type of attribute. Sampling for attributes is applicable when the test yields an answer of yes or no. It answers the question how many.

Two CA-PanAudit Plus routines are available for attribute sampling:

- ATTPCT
- ATTSAMP

ATTPCT calculates what percentage of a file's total records constitutes a representative sample. Calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision (acceptable variance)
- Expected error rate

The accuracy of the sample result can be expressed in terms of these statistical measurements.

ATTSAMP calculates the sample size exactly as ATTPCT but randomly selects the appropriate number of records from the file.

Sampling for Variables

Use the variable sampling routines to estimate the total value of a population composed of items having variable characteristics. For example, the dollar value of an entire inventory is estimated from a sample of that inventory, given normal distribution of the variable in a population. Sampling for variables is applicable when the test evaluates the variable quantities such as dollars, pounds, or days. It answers the question how much.

Two CA-PanAudit Plus routines are available for variable sampling:

- VARPCT
- VARSAMP

VARPCT calculates representative sample size based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Standard deviation

The value of standard deviation can be obtained by using the STDDEV routine.

VARSAMP calculates the sample size exactly as VARPCT but randomly selects the appropriate number of records from the file.

Statistical Validity

The calculations for sample size in the ATTPCT, ATTSAMP, VARPCT, and VARSAMP routines assume there is a normal distribution of items in the population. The calculations comply with the Central Limit Theory for Large Population Distribution Analysis, which indicates that regardless of the distribution of the values in the population, the distribution of items in the sample will tend toward normality as the sample size grows. However, the rapidity with which this tendency occurs depends on the sample size and the relative distribution of the values being sampled. With populations of nonnormal distribution, the samples generated by these routines may not be statistically valid.

If there is a wide variance in the value of items in the population, the stratified sampling routines (STRATIF, STRTEVL) provide more reliable results.

ATTPCT/ATTSAMP

These routines perform an attribute sampling.

- ATTPCT calculates the required sample size.
- ATTSAMP calculates the size and creates a statistically valid random sample.

CAVEVAL

Performs an evaluation of the sample file created by CAVSAMP.

CAVSAMP

Creates a sample file based on a combined attributes and variables proportional sampling algorithm.

DISCPCT/DISCSMP

These routines perform a discovery sampling.

- DISCPCT calculates the required sample size.
- DISCSMP calculates the size and creates a statistically valid random sample.

Proportional Sampling

CA-PanAudit Plus offers three types of proportional sampling routines:

- Dollar unit sampling (DOLUNIT)
- Sampling proportional to size (SPS)
- DOLUNIT and SPS are sampling routines that require extensive professional judgment to determine an appropriate sample size.
- Combined attributes and variables sampling (CAVSAMP) uses a statistical method that the American Institute of Certified Public Accountants (AICPA) recommends. This statistical method calculates a sample size from parameters that you specify.

Each proportional sampling routine uses a different method of achieving a valid statistical sample where the probability that a given physical sampling unit will be in the sample population is proportional to the size of the unit.

Dollar Unit Sampling

Dollar unit sampling specifies the size of the field being examined. These values accumulate until a calculated limit is equaled or exceeded. The record that specifies this limit is written to a sample file. All records above this limit are selected for output to a sample file.

Any negative values processed by the DOLUNIT routine are treated as positive by multiplying the value by (-1). This occurs because the emphasis in dollar unit sampling is on the size of the field being examined, not whether the value contained is positive or negative. If this technique were not employed, encountering a file containing many negative values would delay the occurrence of the limit being exceeded and incorrectly reduce the number of items in the sample file. The dollar unit sampling algorithm is applied to the absolute value of the input values, and totals for both the absolute value and actual file total are presented at the end of the dollar unit report.

For similarities between DOLUNIT and the SPS routine, see Comparison of DOLUNIT and SPS Routine Accumulators later in this chapter.

Dollar Unit Concepts

There are two fundamental concepts important in understanding dollar unit sampling:

- Cell width
- Cutoff

Cell Width

Think of the dollar unit accumulations of an audited field as filling a cell. The value of cell width partially determines what the target value for the current cell will be. The record containing the field that causes the accumulated value to fill the cell is selected for output to the sample file. The cell is increased by a randomized factor of the cell width (a new maximum value is obtained), and the process of filling the cell resumes. Cell width is analogous to the target parameter of the SPS routine.

The selection of a correct value for cell width is important in obtaining meaningful results with DOLUNIT. Generally, there is no rule for selecting an appropriate cell width; however, the following guidelines are suggested:

- Determine the total value of the file
- Subtract from it the total value of all TOP and KEY items (see parameters)
- Divide this by the number of samples desired

This value becomes the cell width, determines the number of cells processed, and subsequently, the number of samples written to the sample file.

Cutoff

The cutoff is a value which, when exceeded by any audited field, causes the record to be written to the sample file. This special sample is called the top stratum and can be directed to a separate file through the use of the TOP parameter.

SPS

The Sampling Proportional to Size (SPS) routine uses an algorithm similar to DOLUNIT to select records for the sample file. With SPS you do not specify a cell width; instead, specify a target value that must be exceeded in order for a record to be selected for sampling.

In DOLUNIT, the value to be exceeded is increased by a randomized factor of the cell width to produce a new upper limit. A record is selected when the accumulated total of the input values exceeds the upper limit. In SPS, the value to be exceeded (target value) never changes. The accumulated total is initialized in a randomized process, and the value of the input record is added to the accumulated total. A record is selected when the accumulated total exceeds the target value. The accumulated total is reduced by the amount of the target value until it is less than the target value.

In DOLUNIT, the accumulated total is always increasing, and the upper limit increments when DOLUNIT selects a record. In SPS, the value to be exceeded is always the same, and the accumulated total decreases when SPS selects a record. As in DOLUNIT, the absolute value of the input field to SPS accumulates so that sampling is not delayed due to a large percentage of negative values.

Comparison of DOLUNIT and SPS Routine Accumulators

DOLUNIT is similar to the SPS routine in that both Routines A-Ccumulate values until a limit (cell width in DOLUNIT and target in SPS) is exceeded, in which case a record is written to a sample file.

In SPS, exceeding the target by amounts greater than the target value has no effect on future sampling. The accumulator is reset to a value less than the target amount every time it is exceeded. In DOLUNIT, exceeding the target value (width in DOLUNIT) by amounts greater than the width has an effect on future sampling. This is because the resetting process does not reset several multiples of target amounts as in SPS; rather, it resets only one multiple of the target amount. Therefore, in DOLUNIT, exceeding the width by large amounts can cause the next record to be selected because of residual accumulation from previous values.

The use of the cutoff parameter in DOLUNIT writes records to an output file if an audited field exceeds the cutoff. When this occurs, the input value is not added to the accumulated value. SPS does not have a cutoff facility.

Generally, DOLUNIT is more versatile than SPS. The logic in the routines is considerably different.

Combined Attributes And Variables Sampling

DOLUNIT and SPS require that a cell width or target value be specified to determine the number of records in the sample file. CAVSAMP performs statistical calculations based on input values to determine a sample size. The sample is determined by calculating a target value from the input values of tolerable error, expected error, and confidence. This target value is identical to the target value in SPS and is used as input to an SPS algorithm that selects the records for sampling.

After the sample file is created, audited amounts must be entered for all recorded amounts in the sample file. This audited sample file is input to the CAVEVAL evaluation routine that reports various conclusions about the population based on the differences found in the sample file. The use of CAVEVAL in association with CAVSAMP is important due to the explicit limitations of this method. See the descriptions of the CAVSAMP and CAVEVAL routines in the chapter "Generalized/Statistical Routines A-C" for an explanation of these limitations. CAVSAMP uses a statistical method that the AICPA recommends.

DOLUNIT

The dollar unit sampling routine selects records for sampling according to monetary units rather than physical attributes.

EACHNTH

This is an interval sampling routine that accepts a starting position and a skip factor. It selects every *n*th record (as identified by the skip factor) until it reaches the end of the input file.

INTSAMP

Randomly selects records for sampling in a specified interval.

POPCOUNT/POPSIZE

Use the population size routines to calculate the population size of a file from within a user-written program or from a stand-alone routine. The primary purpose of the population size routines is to calculate the size parameter in the attribute, variable, discovery, and some of the random sampling routines. Those routines can be executed without knowing the exact population size.

The two CA-PanAudit Plus population size routines are POPCOUNT and POPSIZE.

- POPCOUNT is used with the ATTPCT, VARPCT, and DISCPCT routines.
- POPSIZE is used with the ATTSAMP, VARSAMP, DISCSMP, RANDPCT, and RANDXCT routines.

RANDPCT

This unrestricted random sampling routine creates a sample file by randomly selecting an exact percentage of records from a file.

RANDXCT

This unrestricted random sampling routine creates a sample file by randomly selecting an exact number of records from a file.

Regression Estimation

Regression estimation is a method that uses unstratified variable sampling to estimate total audited amounts.

Method

The regression estimation technique used in CA-PanAudit Plus provides four steps for sampling and analysis that are combined into three separate routines. The steps are:

1. Determine an estimated standard deviation of the original population (REGSAM).
2. Calculate the sample size (REGSAMP).
3. Create the sample file (REGSAMP).
4. Evaluate the selected sample (REGEVAL).

The routines REGSAM, REGSAMP, and REGEVAL comprise the regression estimation method.

Key Terms

The following terminology is used in the documentation to see files in the regression estimation routines:

Original file—This is the original population from which the preliminary sample and the REGSAMP sample are taken.

Preliminary sample—This is the file created for REGSAM to calculate an estimated standard deviation of the original population. It is an unrestricted random sample of the original population.

REGSAMP sample—This is the file created by REGSAMP and is used by REGEVAL to calculate the estimated audited amount for the original population. It is an unrestricted random sample of the original population.

Preliminary Sample

To determine the estimated standard deviation of the original population, a preliminary sample is taken. You should take an unrestricted random sample from the original population (using the RANDXCT or RANDPCT routines). You must establish recorded and audited amounts for this preliminary sample. It must be large enough to contain several nonzero differences between the recorded and audited amounts. You can use attribute tables or attribute sampling to establish the appropriate sample size needed to obtain enough differences.

Then, do the following:

1. Input the preliminary sample file to REGSAM; this establishes an estimated standard deviation of the differences between the recorded and audited amounts.
2. Input the original file to REGSAMP. The estimated standard deviation is used to determine a sample size and creates the REGSAMP sample file. The audited and recorded amounts must be established for this file.
3. Input the REGSAMP sample file to REGEVAL; this calculates the estimated audited amount for the entire file based on the audited amounts and estimated standard deviation.

REGEVAL — Evaluates the sample created in REGSAMP. This is the third and final step in regression estimation.

REGSAM — Determines the estimated standard deviation of the regression population. This is the first step in regression estimation.

REGSAMP — Determines the sample size and creates the sample file for the regression estimate. This is the second step in regression estimation.

SPS — The sampling proportional to size routine creates a sample where the probability of the selection of a record is proportional to the size of the record.

STOPORGO — Selects multiple random samples from a single file.

Stratified Random Sampling

Stratified sampling is most successfully applied to a population where a wide variance or uneven distribution of values is to be measured. Other techniques, such as random variable sampling, cannot provide reliable results in this situation due to the irregular distribution of values.

In stratified sampling, values are separated into classes (strata) that contain items with reasonably similar values. Each stratum is treated as a separate population, and its sample size is determined independently of other strata. However, in a stratum, the probability that any one item is included in the sample is the same as for any other item in the stratum.

This sampling method often results in increased efficiency by decreasing the necessary sample size. It is used to pay special attention to portions of the file of particular interest through the practical stratification of the file.

Target Value

CA-PanAudit Plus stratified sampling routines give you the flexibility of choosing the multiple stratum boundaries.

On multiple targets, until the end record value is reached, the current target is used. When the end record value is reached, the current target is finished, and the new target is started.

The target value controls processing in the following way. The records in the file are sorted in ascending sequence. The values are accumulated until the target value is exceeded. At this point, a stratum is full, and a new stratum is formed in the same manner.

Consider the following example:

```
          STRATUM SIZE: 2700000 MATERIALITY:          50,000.00
           PRECISION:      1,000,000.00  CONFIDENCE: 95 %
                                                      STD    SAMP    PCT
 FROM            TO       FREQ        TOTAL           DEV    SIZE   SAMPLE
       .00          .00                       .00        .00           .0
       .01       703.45     6,648    2,700,168.00     175.29    19      .2
    703.45       988.03     3,201    2,700,811.77      82.25     4      .1
    988.03    13,929.32       483    2,705,112.05   4,340.24    34     7.0
 13,929.32    19,754.30       159    2,705,434.23   1,675.18     4     2.5
 19,754.30    39,947.50        86    2,713,248.20   5,743.39     8     9.3
 39,947.50    50,000.00        58    2,610,647.50   2,945.18     3     5.1
```

The file has been separated into strata based on the specified target value of 2,700,000. All members in an individual stratum are related in that they are relatively similar in value.

The specific items for each stratum are:

- Starting value (FROM)
- Ending value (TO)
- Standard deviation
- Total value

The total value is always greater than or equal to the target value. Each successive stratum contains fewer records than the previous stratum because the value of the field is increasing, and fewer items are required To exceed the target value.

For an ideal stratification to occur, it is best to select a target value that results in an excellent separation of values in each stratum. Although it is impossible to achieve a perfect stratification. You can use the INTERVL routine prior to selecting the target value to study the distribution of the file to aid in selecting a good target value. When you select a target value, consider the following:

- Try to construct each individual stratum with a minimum of approximately 50 items. When the frequency falls below 50, the accuracy of the sample for that stratum rapidly deteriorates. The accuracy of the results obtained from the STRTEVL routine also decreases.
- A good criterion to use in calculating the target value is to first choose the materiality. All values that exceed materiality are separated into the top stratum for a 100 percent sampling. Calculate the total value of all items and subtract the amount of the values in the top stratum. Divide this by the number of stratum desired. This figure becomes the target value.
- The number of strata desired must not be very large. The more strata that exist, the less benefit is derived in attempting to decrease the sample size while maintaining specified precision and confidence. The number of strata depends on the range and number of items in a file, so an absolute ideal number cannot be chosen. However, the use of five to ten strata is generally considered to be acceptable.
- After choosing a target value, the results may show that the stratum immediately before the last stratum is less than full (for example, the stratum total is considerably less than the target value). If the stratum total is very low (for example, less than 50 percent of the target value), the sample size for this stratum is adversely affected because one of the factors used to determine sample size is frequency. This can cause sample sizes for this stratum to decrease to zero, which is undesirable, because all strata must contribute to the sample file with an absolute minimum of one, and preferably at least two, records. This situation also affects the accuracy of the results of the STRTEVAL routine.

  To reduce any adverse effects of not filling the next-to-last stratum, the recommended procedure is to change the target value up or down, so this stratum total is at least 90 percent of the target value. If the target value is increased, the partially filled stratum is eliminated and absorbed into previous strata. If the target value is decreased, the partially filled stratum becomes filled to an acceptable level (90 percent of target value).

  Changing the target value up or down to fill the next-to-last stratum changes all of the stratum statistics, including sample size. Neither result is more correct than the other because both result in valid sample files. Your choice can be made on the basis of the evenness of the stratification, a desired number of strata, or for other statistical reasons.

Work Files

A work file must be defined in the JCL when utilizing either STRATIF or STRTEVL. The work file is named STRTBL and must be specified as the DD (or DLBL) name on an appropriate JCL statement. The stratified sampling routine, STRATIF, creates the file, and STRTEVL uses the same file as input. JCL examples are provided for the STRATIF and STRTEVL routines in the chapter "Generalized/Statistical Routines S-Z."

STRATIF — This routine creates a stratified random sample and allows variable size strata.

STRTEVL — This routine evaluates the sample created by STRATIF.

VARPCT/VARSAMP — These routines perform a variable sampling.

- VARPCT calculates the required sample size.
- VARSAMP calculates the size and creates a statistically valid random sample.

Statistical Forecasting Routines

The CA-PanAudit Plus statistical forecasting routines use regression analysis as the basis for their statistical forecasting. All statistical forecasting routines are stand-alone CA-PanAudit Plus routines. For a detailed description of how to use stand-alone routines, see the chapter "Advanced Techniques."

The statistical forecasting routines are:

MULTREG — This regression analysis routine solves the regression equation for two or three independent variables and provides analyses of variance and correlation.

SIMPREG — This regression analysis routine solves the simple regression equation and provides a correlation analysis.

Chapter 3

Coding Guidelines

This chapter contains general instructions and guidelines for using the CA-PanAudit Plus system.

This chapter is an instructional module of special importance to new users of CA-PanAudit Plus and persons having no experience with CA-Easytrieve Plus, the host language. All material presented later in this guide presupposes a knowledge of the fundamentals given in this chapter.

The following topics are included:

- Types of CA-PanAudit Plus Routines (Inline and Stand-alone)
- Screening of Input Data With the Stand-alone Routines

Types of CA-PanAudit Plus Routines

There are two basic types of CA-PanAudit Plus routines:

- Inline
- Stand-alone

These two types of routines are used differently in CA-PanAudit Plus and require you to supply different CA-Easytrieve Plus statements.

Overall differences between the routine types are listed in the following table and discussed in the topics that follow. In general, inline routines function in many different ways, while stand-alone routines are more consistent in their format. The following guidelines do not apply to the SMF Routines, the Graphing facility, or the Job Information Facility (JIF).

Inline	Stand-alone
Only one macro name is required to invoke a routine.	Generally requires two macros to invoke one routine.*
An open routine that can be combined with other routines.	A complete system that defines input files, performs the algorithm, and produces a report.
You must code a JOB statement and any required report subactivity.**	You must supply the library section.
Parameter list never contains an input file.	Parameter list always contains an input file.
In some cases you must test internally defined fields to determine action or whether to print a line of a report; in other cases, results of calculations are placed in a user-supplied field.	Creates some form of report, and in some cases, an output file.

* There are exceptions. Some stand-alone routines operate with one invocation only, and some require three macros. For further information, see the detailed descriptions for each specific routine in later chapters.

** Report subactivities are discussed in the CA-Easytrieve Plus *Reference Guide.*

Inline Routines

Inline CA-PanAudit Plus routines require only one macro invocation. They require varying degrees of user coding. This chapter explains the code that is standard for all inline routines. The documentation of the individual inline routines gives the specific coding you must supply.

This individual coding usually consists of testing a specially named internal field for a YES or NO condition. Based on the value in this internal field, appropriate action is then taken.

Inline routines contain the following sections:

- Environment section (optional and rarely required)
- Library section (optional)
- JOB statement (required)
- Optional user code
- CA-PanAudit Plus routine
- Optional user code

```
FILE . . . .
FILE . . . .

JOB. . . .
                            (language statements)
                             inline routine
                            (language statements)

REPORT. . . .
```

Aspects of the library section specific to inline routines are found next, together with an explanation of the optional user codes.

Library Section

If your program does not use an input file, you can define working storage fields in the activity section of the job and omit the library section. However, if an input file is required, you must code the library section in your program.

Optional User Code (Prior to Routine)

The optional user code, prior to invoking a CA-PanAudit Plus routine, consists of any code you want to include before invoking the routine. This can consist of field definitions, other CA-PanAudit Plus routines, user-defined macros, and CA-Easytrieve Plus statements.

You must be careful to follow CA-Easytrieve Plus coding guidelines for any statements coded in this section. For example, report subactivities are not allowed in this section.

CA-PanAudit Plus Routine

You invoke an inline CA-PanAudit Plus routine as you would a macro, coding a percent sign (%) followed by the name of the routine and any applicable parameters as shown in the following example:

```
%routine-name parm1 parm2 ... parmn
```

Routine names and syntax of this statement are fully described in later chapters.

Optional User Code (Following Routine)

This optional section consists of any code you want to include after the CA-PanAudit Plus routine is finished. It can consist of testing internal flags of the routine, another CA-PanAudit Plus routine, or any valid CA-Easytrieve Plus statements. You must be careful to follow all CA-Easytrieve Plus coding guidelines for any statements coded in this section.

If an inline routine is coded using the JOB INPUT NULL statement, a STOP statement must be coded to terminate execution of the job (see Example One shown next).

Examples

The following are three examples of inline CA-PanAudit Plus routines.

Example One

```
JOB INPUT NULL
%ATTPCT 100000 90 1.8 3.2
%ATTPCT 100000 95 1.8 3.2
%ATTPCT 100000 98 1.8 3.2
STOP
```

This example is a simple form of an inline routine. It demonstrates the use of the ATTPCT statistical routine to evaluate the effects on sample size of varying the confidence factor.

The ATTPCT routine calculates what percent of a file's total records constitute a representative sample. In this job, ATTPCT is used to evaluate how many records from a sampling of 100,000 records constitute a representative sample at 90 percent, 95 percent, and 98 percent confidence levels. User-specified precision and error factors follow the confidence level.

Each invocation of ATTPCT prints a report of the input parameters and the calculated sample percentage. The job contains only a JOB statement, which states there is no input file, three invocations of the ATTPCT routine, and a STOP statement.

Example Two

```
FILE PAYFILE
  EMP-NUM        22   9  N
  HIRE-DATE      31   5  N
JOB INPUT PAYFILE
%DATEVAL HIRE-DATE YYDDD
IF DATEVAL-FLAG  EQ 'NO'
  DISPLAY EMP-NUM HIRE-DATE
END-IF
```

This example demonstrates the use of the generalized routine, DATEVAL, to print a report of any hire dates in the payroll file that are not in the format YYDDD. DATEVAL tests the content of a specified date field, HIRE-DATE, and sets a flag for those not in the specified format.

This job contains a library section, a JOB statement, a CA-PanAudit Plus routine, and optional user code.

- The JOB statement indicates that PAYFILE, described in the library section, is the input file.
- The optional user code (the IF statement) tests the internal field DATEVAL-FLAG, which is defined in the DATEVAL routine. If the value is NO, the employee number and invalid date are printed.

Example Three

This example shows a more complex use of an inline routine. It shows multiple CA-PanAudit Plus routines, optional user code before and after the routines, and CA-Easytrieve Plus reports.

```
FILE CUSTFIL
  ACCOUNT-NUM       23   5  N
  ACCOUNT-NAME      47  20  A
  INVOICE-DATE      84   6  N
  INVOICE-AMT       95   6  P  2
JOB INPUT CUSTFIL
IF INVOICE-AMT EQ 0
  GO TO JOB
END-IF
%DATEVAL INVOICE-DATE MMDDYY
IF DATEVAL-FLAG  EQ 'NO'
  PRINT BAD-DATE-REPORT
ELSE
  %DAYSAGO INVOICE-DATE MMDDYY GT 30
  IF DAYSAGO-FLAG EQ 'YES'
     PRINT OVERDUE-REPORT
  END-IF
END-IF
REPORT BAD-DATE-REPORT
TITLE 1 'INVOICES WITH INVALID INVOICE DATES'
LINE ACCOUNT-NUM ACCOUNT-NAME INVOICE-DATE
REPORT OVERDUE-REPORT
TITLE 1 'INVOICES OVER 30 DAYS OLD'
LINE ACCOUNT-NUM ACCOUNT-NAME INVOICE-DATE INVOICE-AMT
```

- This job begins with a library section and a JOB statement that defines the input file as CUSTFIL.
- Optional user code before the invocation of a CA-PanAudit Plus routine checks for zero invoice amounts. If the amount is zero, the record is bypassed from processing by the statement GO TO JOB which, in effect, passes control to the JOB statement and reads another input record.
- The generalized routine DATEVAL checks that the invoice date is valid. If it is invalid, DATEVAL prints a line of the BAD-DATE-REPORT, defined at the end of the program. If the date is valid, the generalized routine DAYSAGO is used to check if the invoice is over 30 days old. If it is overdue, DAYSAGO prints a line of the OVERDUE-REPORT, defined at the end of the program.

Stand-alone Routines

Stand-alone CA-PanAudit Plus routines are complete systems in that they require an input file, perform specific processing, and generate a listing and possibly an output file. Stand-alone routines are more structured than inline routines and follow a standard format. However, in this standard format, there are two distinct types of stand-alone routines, each using different types of CA-Easytrieve Plus processing. You can invoke both types of stand-alone routines in an identical manner in most applications. In some sophisticated applications, you may want to use the processing capabilities of both types of stand-alone routines.

Stand-alone DISPLAY and Stand-alone REPORT Routines

The two types of stand-alone routines are DISPLAY and REPORT. Their names come from the CA-Easytrieve Plus statements that print listings. In CA-Easytrieve Plus , a DISPLAY statement prints lines of output but does not perform any special functions. The REPORT statement prints lines of output and can perform automatic page numbering, titling, footing, control breaks, summing, and other special functions.

This guide uses the term stand-alone to denote both DISPLAY and REPORT type routines and that the information applies to both types. If the information is specific to one of the types, then the guide will use the term stand-alone DISPLAY or stand-alone REPORT.

The following table shows the different places in your programs and routines where you must code these two types of output statements:

DISPLAY Statements	REPORT Statements
You must code DISPLAY statements as inline statements.	A REPORT statement is a subactivity and must appear at the end of a program.
Display statements can appear almost anywhere in a CA-PanAudit Plus program.	A stand-alone REPORT routine is, in effect, closed off by the existence of the REPORT statement at the end of the routine.
A stand-alone DISPLAY routine is open in nature, and you can code additional statements after a stand-alone DISPLAY routine.	The type of code that can occur after the report statement is limited.

See the CA-Easytrieve Plus *Reference Guide* for details regarding the DISPLAY and REPORT statements.

Note: The chapter "Advanced Techniques" explains the different processing options for DISPLAY and REPORT stand-alone routines and contains examples illustrating how you can use each type.

Logic-After-Invocation

You do not require knowledge of the different processing options available with the DISPLAY and REPORT types, unless you are using advanced processing techniques. The types of stand-alone routines presented in subsequent chapters are labeled in the Operation section. If your use of the routine does not involve any logic after the invocation of the routine, then you do not need a detailed understanding of the information on stand-alone DISPLAY and stand-alone REPORT routines as detailed in the chapter "Advanced Techniques."

Stand-alone Routine Format

The structure of a stand-alone routine differs from the structure of an inline routine; therefore, the two types of routines have different mandatory and optional sections. In a stand-alone routine, you must specify the library section and the CA-PanAudit Plus routine (first invocation). There are also optional sections in a stand-alone routine.

Stand-alone routines contain the following sections:

- Environment section (optional and rarely used)
- Library section (required)

- Logic-before-invocation (optional)
- CA-PanAudit Plus routine (first invocation, required)
- Screening logic (optional)
- CA-PanAudit Plus routine (second invocation - if required)
- Logic-after-invocation (optional)

Library Section

All stand-alone routines require a library section. The library section must describe the input and optional output files that the macro invocation specifies. You can define working storage fields either in the library section or in the activity section of your program (see Screening Logic on the following page).

Logic-Before-Invocation

The logic before the invocation of the first macro must consist of an entire CA-Easytrieve Plus activity or group of activities. This includes:

- JOB activities
- SORT activities
- REPORT subactivities

Use an activity defined in this section to screen the input file or calculate data, such as a population size, before invoking the CA-PanAudit Plus routine. You can also use an activity defined in this section to print a report prior to the execution of the routine.

The logic-before-invocation section is optional and is most often used in sophisticated applications. For details and examples of logic-before-invocation, see the chapter "Advanced Techniques."

CA-PanAudit Plus Routine (First Invocation)

For most CA-PanAudit Plus stand-alone routines, you must invoke two macros to execute the routine. This is done to allow you to screen the input file and code any CA-Easytrieve Plus logic before the routine begins processing. This optional screening logic (discussed next) is placed between the two invocation statements for the routine.

Invocation of the first CA-PanAudit Plus stand-alone routine consists of a percent sign (%) followed by the name of the routine and the character 1, followed by any applicable parameters as shown in the following example:

```
%routine-name1 parm1 parm2 ... parmn
```

Routine names and syntax are fully described in later chapters.

Screening Logic

One of the processing options of CA-PanAudit Plus is the ability to screen input data. The screening of the input file allows you to examine fields in the input record and eliminate records from processing. It also allows you to perform complex processing and decision making.

The use of the screening facility is not required for the successful execution of any CA-PanAudit Plus stand-alone routine.

Complex processing requires an indepth knowledge of CA-Easytrieve Plus and is not discussed in detail here. Example Three — Complex Screening Logic, later in this chapter, demonstrates complex screening logic.

Screening logic is coded between the macro invocation statements. You code screening logic after the first invocation of the macro.

For example, if you want records bypassed from processing, you must do the following:

- Code the invocation of the first macro.
- Code your screening logic and use the CA-Easytrieve Plus branching statement GO TO JOB.

  This causes control to be passed to the JOB statement which, in effect, bypasses the record from processing and reads the next record from the input file.
- Code the invocation of the second macro.

Example Two and Example Three, later in this chapter, contain screening logic.

Some stand-alone routines do not allow screening. This is because unpredictable results may occur in these routines if screening processing is allowed. Any routine that requires the invocation of two macros will allow input file screening. Stand-alone routines that require only one macro invocation do not allow screening.

CA-PanAudit Plus Routine (Second Invocation)

The second invocation of the routine consists of a percent sign (%) followed by the name of the routine and the character 2, and usually no parameters as shown in the following sample. The only parameters specified on the second invocation are the options that control output files and report subactivities.

```
%routine-name2 [parm1 parm2 ... parmn]
```

The second invocation is required for all stand-alone routines that allow screening of input files.

Logic-After-Invocation

Logic-after-invocation is an optional section of code that sophisticated applications often use. Logic-after-invocation follows the final macro of a stand-alone routine and can consist of different CA-Easytrieve Plus statements depending on whether the routine is a DISPLAY or REPORT type.

The following example is a DISPLAY type:

```
FILE . . . .

FILE . . . .

Stand-alone routine . . . .
(language statements)
(screening code)
DISPLAY . . . .
```

The following example is a REPORT type:

```
FILE . . . .

FILE . . . .

Stand-alone routine . . . .



REPORT. . . .
```

The logic coded in this section most often consists of statements which the screening code section performs. For details and examples of both types of stand-alone routines (DISPLAY and REPORT), see the chapter "Advanced Techniques."

You can also use logic-after-invocation for REPORT procedures. For details and an example of this application of stand-aloneroutines (DISPLAYand REPORT), see the chapter "Advanced Techniques."

Examples

The following are three examples of stand-alone CA-PanAudit Plus routines.

Example One — Without Screening Logic

```
FILE CUSTFIL
  BALANCE    1 10  P  2
  MONTH     11  2  N
%VERSUS1 CUSTFIL BALANCE MONTH GRAPH 0 2
%VERSUS2
```

This example demonstrates the use of a stand-alone routine. It contains the mandatory library section for the input file, CUSTFIL. The VERSUS routine contains two macros, but no screening logic is performed.

Note: The second macro has no parameters.

In this particular example, the distribution analysis routine VERSUS is used to show the frequency distribution of values from the field BALANCE in a specified month. The user has asked for a graph as part of the output report.

Example Two — With Screening Logic

```
FILE CUSTFIL
  BALANCE     1 10  P   2
  MONTH      11  2  N
%VERSUS1 CUSTFIL BALANCE MONTH GRAPH 0 2
IF BALANCE LE 0
  GO TO JOB
END-IF
%VERSUS2
```

This example is identical to the first except that logic to screen input records is included between the invocations of VERSUS1 and VERSUS2. VERSUS performs its calculations properly whether screening logic is present or not.

The effect of the screening logic in this example is to bypass all records with a balance less than or equal to zero. This causes the output produced by VERSUS to be a report of only positive balances.

Generally, screening logic consists of three statements:

- An IF statement
- A GO TO JOB statement
- An END-IF statement

If the END-IF statement is missing, a CA-Easytrieve Plus syntax error is detected, and a diagnostic message is printed. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*.

Example Three — Complex Screening Logic

```
FILE PAYFILE
  EMPLOYEE-CODE     5   1   A
  GROSS-PAY        23   5   P   2
  HIRE-DATE        31   5   N
FILE SAMPFIL
%DOLUNIT1 PAYFILE GROSS-PAY 10000 2000 13453
IF EMPLOYEE-CODE EQ 'P'
  GO TO JOB
END-IF
%DAYSAGO HIRE-DATE YYDDD LT 30
IF DAYSAGO-FLAG EQ 'YES'
  GO TO JOB
END-IF
%DOLUNIT2 SAMPFIL
```

This example demonstrates more complex screening logic.

- The input and output files are defined.
- The first invocation of the DOLUNIT routine is coded.

DOLUNIT selects records for sampling according to monetary units rather than physical attributes.

The screening logic consists of two parts:

- The first part bypasses all employees with a code of P (part-time employees).
- The second part invokes the inline CA-PanAudit Plus routine DAYSAGO.

  DAYSAGO is a generalized routine that determines the number of days between the current date and a specified date (HIRE-DATE). If the difference between the two dates is less than 30, DAYSAGO-FLAG is set to the value YES. The following IF statement tests for this condition and executes a GO TO JOB if the flag is YES. This bypasses the processing of all employees hired in the last 30 days.
- The second DOLUNIT macro, DOLUNIT2, is coded. The parameter specifies the output file to be written to by the DOLUNIT routine.

Chapter 4

Generalized/Statistical Routines A-C

This chapter lists alphabetically, and gives detailed descriptions of, routines ADDRCMP through CONVEA.

ADDRCMP

The ADDRCMP routine produces a report of potentially duplicate street addresses in a boundary file, usually defined as the ZIP code. Standard keys are created for each address to account for different representations of the same address. For example, P.O. Box #54 and POST OFFICE BOX 54 have the same standard key of POBOX54, and are flagged by ADDRCMP as duplicate addresses. A boundary field is defined to eliminate the flagging of duplicate addresses in different areas, such as cities, states, or ZIP codes.

Syntax

```
%ADDRCMP1  infile  [LRECL length]
%ADDRCMP2  infile     field boundary {outfile} {field2 field3}
                                     {NOFILE } {SUMMARY      }
```

`infile`

Specify the name of the input file to ADDRCMP. A valid name is any previously defined file.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of disk-storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 39 work bytes + 4 RDW bytes = LRECL
```

`field`

Specify the field for which duplicate street addresses are searched. If the address in this field is the same in two or more records, a duplicate exists. A valid name is any field in infile.

`boundary`

Specify the boundary field in the infile input file for duplicate address checking. This is usually the ZIP code field. This causes ADDRCMP to check for duplicate addresses only in this boundary and eliminates the flagging of duplicate addresses in other areas.

```
{outfile}
{NOFILE }
```

Specify whether records with duplicate addresses are to be written to an output file:

outfile–Specify the file to which the duplicate records selected are written. File characteristics must be coded on the FILE statement for this output file. The file specified on outfile must have the same file characteristics as the input file. Valid names for outfile include any previously defined file. If you want to include the standard address key created by ADDRCMP, increase the outfile size by 38 and add the following field definition to the outfile definition:

```
ADDRCMP-KEY   start   38   A
```

where start is the length of infile plus one.

The standard address key is placed in the ADDRCMP-KEY field.

NOFILE–Specify this option if records with duplicate addresses are not to be written to an output file.

```
{field2 field3}
{SUMMARY      }
```

Specify whether the report contains a detail line for each duplicate record or is a summary report giving the total of duplicates.

field2 field3–When this option is specified, three fields are listed for each duplicate record:

- Field (the field for which duplicates are searched)
- Field2
- Field3

Valid names are any previously defined fields.

SUMMARY–When this option is specified, a summary report is produced. This consists of the total number of sets of duplicates and the total of all duplicate records in the file.

Operation — Stand-alone REPORT

Use the ADDRCMP routine to get either information on duplicate street addresses or a summary of duplicate street addresses.

Operation — Database

ADDRCMP can access database and nondatabase files without changes in specification parameters. The infile parameter can either be a nondatabase file name or the name of a database file defined in the library section.

Examples

The following are two examples of the ADDRCMP routine.

Example One

In this example, NAME and ACCNT are used for ID fields, and the detail report is specified.

Input

```
FILE ADDRFILE F(80)
   NAME       1 18 A
   ADDRESS   19 30 A
   CITY      49  5 A
   STATE     54  2 A
   ZIP       56  5 N
   ACCNT     62  8 N
 %ADDRCMP1 ADDRFILE
 %ADDRCMP2 ADDRFILE ADDRESS ZIP NOFILE NAME ACCNT
```

Output

```
                                    REPORT OF DUPLICATE ADDRESSES
                              INPUT FILE: ADDRFILE    BOUNDARY: ZIP    FIELD: ADDRESS
                                          NOFILE IS PRODUCED

                                                                                                NUMBER OF
ZIP       NORMALIZED KEY                ACTUAL ADDRESS               NAME              ACCNT    DUPLICATES

11111 POBOX111                          P O BOX111                   JACK  DOE         00066666
                                        P O BOX 111                  JOSEPH KOOL       00000000
                                        P. O. BOX 111                PETER GUNN        00077777
                                        P.O. BOX 111                 WILD BILL HAYCOCK  00011111
                                        POST OFFICE BOX 111          WILLIAM SCOTT     00055555
ADDRCMP-KEY TOTAL                                                                                        5

11111 RR2BOX69                          R.F.D. 2, BOX 69             BOB JOHNS         00155554
                                        R.F.D. #2 BOX 69             HERB LEWIS        00166665
                                        R R 2 BOX 69                 JANE ROBERTS      00099999
                                        R.R. 2, BOX 69               PETER PAUL        00111110
                                        RURAL ROUTE 2 BOX 69         SCOTT JONES       00122221
                                        RFD 2, BOX 69                TOM PATTERSON     00133332
                                        RR 2, BOX 69                 WILBUR SMITH      00088888
ADDRCMP-KEY TOTAL                                                                                        7

11111 1234SUNSHINE                      1234 SUNSHINE STREET         JANE BABBITT      00188887
                                        1234 SUNSHINE PLACE          JESSIE SMITH      00199998
ADDRCMP-KEY TOTAL                                                                                        2

ZIP TOTAL                                                                                               14

22222 HCR5BOX9                          HIGHWAY CONTRACT ROUTE 5 BOX 9 JOHN MARIO       00555550
                                        HCR #5, BOX 9                  ROBERTO NUNEZ    00577772
ADDRCMP-KEY TOTAL                                                                                        2

22222 456NALASKA                        456 N. ALASKA STREET           JANE DOE         00488884
                                        456 NORTH ALASKA STREET        UUG JOHNS        00466662
ADDRCMP-KEY TOTAL                                                                                        2

ZIP TOTAL                                                                                                4

40372 HCR5BOX9                          HCR #5, BOX 9                  BARNEY RUBBLE    01211160
                                        HCR #5, BOX 9                  MIGUEL TORRES    01090044
                                        HIGHWAY CONTRACT ROUTE 5 BOX 9 PAUL SMITH       01170788
                                        HCR #5, BOX 9                  SAMSON JONES     01130416
ADDRCMP-KEY TOTAL                                                                                        4

ZIP TOTAL                                                                                                4

FINAL TOTAL                                                                         22
```

Example Two

In this example, SUMMARY is specified, and an outfile is produced. A subsequent JOB activity reports the contents of the output file.

Input

```
FILE ADDRFILE F(80)
  NAME       1 18 A
  ADDRESS   19 30 A
  CITY      49  5 A
  STATE     54  2 A
  ZIP       56  5 N
  ACCNT     62  8 N
FILE OUTFILE F(80)
%ADDRCMP1 ADDRFILE
%ADDRCMP2 ADDRFILE ADDRESS ZIP OUTFILE SUMMARY
```

Output

```
                   SUMMARY REPORT OF DUPLICATE ADDRESSES
          INPUT FILE: ADDRFILE    BOUNDARY: ZIP    FIELD: ADDRESS
                            OUTFILE IS PRODUCED


                                       TOTAL NUMBER OF
                                      DUPLICATES IN ZIP

                     ZIP TOTAL                14
                     ZIP TOTAL                 4
                     ZIP TOTAL                 4
                     FINAL TOTAL              22
```

AGING

The AGING routine calculates the age of a record, places it in a specified age category, and prints a report based on the age of records in the input file.

To calculate the age of a record, AGING uses the current date or a date you specify in the BASEDATE parameter. The date in the input record is subtracted from the current or base date to compute the age. A report is produced listing an amount field and two optional identification fields. The format of the report is controlled by seven optional keyword parameters.

By independently performing your own aging analysis using CA-PanAudit Plus, you can:

- Verify that proper aging is being performed
- Verify file total against the General Ledger
- Identify potential problem areas

Syntax

```
%AGING1 infile datefield format1 amount [BASEDATE date]    +
        [BASEFORM format2] [NUMDIGITS number]              +
        [ DAYS  {  OLD    } ]                              +
        [       { OVERDUE } ]

         [RANGE   days] [RANGE1 days] [RANGE2 days]        +
         [RANGE3  days] [RANGE4 days] [RANGE5 days]        +
         [RANGE6  days] [RANGE7 days]

%AGING2 [idfield1 [idfield2] ][CURANGE] [INTERVALS number] +

        [REPORTTYPE { DETAIL  } ]
        [           { SUMMARY } ]
```

`infile`

Specify the name of the input file to AGING. A valid name is any previously defined file.

`datefield`

Specify the name of the datefield on which the aging analysis is based. The date in this field must be in the format specified by format1. A valid datefield name is any field defined in the input file.

`format1`

Specify the format of datefield. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of datefield is not checked for a valid date with the specified format. If you want date validation, use the DATEVAL routine before coding AGING. The only valid Julian format is YYDDD.

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`amount`

Amount is the name of a numeric field that is totaled, and controlled in the printed report by the BASEDATE, RANGE, INTERVALS, CURANGE, and REPORTYPE parameters. A valid value is any numeric field defined in the input file.

```
[BASEDATE date]
```

This optional parameter specifies the date used in calculating the age of records. The date in datefield is subtracted from this date to compute the age of a given record.

A valid value for date is an actual numeric value or the name of a working storage field containing the value. The date in this field must be in the format specified in format2. The default value for date is the current system date.

```
[BASEFORM format2]
```

BASEFORM is an optional parameter, and the default value is MMDDYY. BASEFORM supplies the format of BASEDATE and is used only if BASEDATE is not in the MMDDYY format. If you do not code BASEDATE, do not code BASEFORM. If BASEDATE is not specified, BASEFORM defaults to the format of the system date and must not be coded. If BASEDATE is specified, BASEFORM can be coded, or it can be omitted and defaults to MMDDYY.

Specify the format for BASEDATE as shown in the format1 parameter.

```
[NUMDIGITS number]
```

This optional parameter is used when the report generated by AGING overflows the print line. When this occurs, diagnostic message B061 is generated, stating the number of print positions that have overflowed.

For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*. You can specify this parameter to reduce the default number of print positions allowed for each range on the AGING report. The default value is 11. A valid value for number is an actual numeric value ranging from 2 to 11.

Note: You do not need to specify this parameter if the print line does not overflow. See Operation — Stand-alone REPORT in this routine for additional information.

```
[DAYS{OLD    } ]
[    {OVERDUE} ]
```

Use this option to specify the method that AGING uses to categorize the age of a record. Two methods are available:

OLD—This specifies a categorization where the data being aged is always a date in the past, for example, a shipping date or invoice date. In this method of AGING, the age of a record (basedate minus specified date) is always greater than zero (assuming current date as basedate). This method of AGING provides an aging analysis based on a record being a calculated number of days old. OLD is the default value.

Using this method of AGING, a current record is one with an age greater than or equal to zero and with an age less than the value that the RANGE days parameter specifies. A RANGE1 record has an age greater than days and less than or equal to two times the value of days.

For example, if RANGE is 30, and the age of a record is 45 days, the current period is defined as zero to 30 days old, and RANGE1 is defined as 31 to 60 days old. Therefore, the record falls into the RANGE1 category. See Example One for sample input and output.

OVERDUE—This specifies a categorization where the date being aged can be a date in the future, for example, the next payment date for a loan. In this manner, the age of a record (basedate minus specified date) can be less than zero (assuming current date as basedate). This method of AGING provides an aging analysis based on a record being a calculated number of days overdue.

Using this method of AGING, a current record is one with an age of less than or equal to zero. A RANGE1 record has an age greater than zero and less than or equal to the value of days.

For example, if RANGE is 30, and the age of a record is 45 days, the current period is defined as less than or equal to 0 days old (such as not overdue), RANGE1 is defined as one to 30 days overdue, and RANGE2 is defined as 31 to 60 days overdue. Therefore, the record falls into the RANGE2 category. See Example Two for sample input and output.

`[RANGE days] ... [RANGE7 days]`

These optional parameters establish the time interval covered by each category in the report. The time interval is specified in days. The default for RANGE is 30 days and for RANGE1...RANGE7 there is no default. A valid value for days is an actual numeric value or the name of a field containing the numeric value.

RANGE can be used by itself to generate fixed-length ranges. RANGE1...RANGE7 can be used to generate ranges of varying lengths.

The rules for determining the number and type of these ranges are as follows:

- If the value of INTERVALS is greater than the number of time intervals specified by the RANGE...RANGE7 parameters, additional time intervals are created. The number of additional time intervals created is the amount of the difference of the value of INTERVALS and the number of RANGE...RANGE7 parameters specified. The length of time used for each of these intervals is that of the last time interval specified via the RANGE...RANGE7 parameters.
- If the value of INTERVALS is less than the number of RANGE...RANGE7 parameters, the RANGE...RANGE7 parameters after the value of INTERVALS will be ignored, and processing continues.

When a particular RANGE...RANGE7 value is specified for AGING, all preceding RANGE...RANGE7 values must be specified on the same macro invocation statement.

The following is a valid usage of the RANGE...RANGE7 parameters because RANGE, RANGE1, and RANGE2 are all specified in a consecutive, ascending sequence.

```
%AGING1  CUSTFIL DATE MMDDYY BALANCE RANGE 30 RANGE1 25 RANGE2 21
```

```
[idfield1 [idfield2]]
```

Idfield1 is an optional parameter that names an informational field used to sequence the input file. All records with identical values for idfield1 are grouped and aged by date ranges according to the value of the base date minus the value of datefield. A valid value is any previously defined field.

Idfield2 is an optional parameter that names an informational field that is printed for identification purposes on the aging report. A valid value is any previously defined field. If idfield2 is specified, idfield1 must be specified.

```
[CURANGE]
```

The optional CURANGE parameter determines if current records can appear on the report. Current records are those of which age is less than or equal to the number of days specified in the RANGE parameter.

For example, if RANGE30 is specified, current items are records of which age is less than or equal to 30 days. If CURANGE is specified, current records appear on the report. If CURANGE is omitted, current records do not appear on the report. The number of current records omitted and their totals appear separately at the bottom of the report. The category for current records is in addition to the number of categories specified in the INTERVALS parameter.

```
[INTERVALS number]
```

This optional parameter establishes the number of time intervals printed on the report. The number of intervals is independent of the current range explained previously. Valid values are the actual numeric values from 1 through 8.

For example, if the RANGE value is 30 (30 days), an INTERVALS value of 1 creates one category of over 30 days. A value of 2 causes two categories of 31 to 60 days and over 60 days. The default value for INTERVALS is 3. For time intervals of varied lengths, see the explanation of the RANGE...RANGE7 parameter for usage of the INTERVALS parameter.

```
[REPORTYPE{DETAIL }]
[         {SUMMARY}]
```

This optional parameter indicates whether a detail report or summary report is produced. The default is DETAIL.

DETAIL—Three fields from each record are listed on the report:

- Amount
- Optional field idfield1
- Optional field idfield2

DETAIL is the default value.

SUMMARY—Only a summary report, controlled by idfield1, is generated.

Operation — Stand-alone REPORT

The input file and the report are sorted in ascending order by idfield1, datefield, and idfield2. All items with identical values for idfield1 have the amount field summarized and totaled according to the aging of datefield.

The AGING routine contains only four mandatory parameters. If only the mandatory parameters and the optional CURANGE parameter are specified, AGING produces a report with a base date of the current system date, four intervals consisting of the current category and three categories of 30 days, and the detail lines. This is, in effect, a standard aging with categories of 0-30, 31-60, 61-90, and over 90 days. AGING can be invoked using only five parameters to produce a standard aging report shown in Example One.

Optional parameters allow for control of the base date, the time intervals of the categories, the number of categories, and the inclusion or exclusion of the current category and detail lines. Any number of these parameters can be included or excluded because each has a default value. Use BASEFORM only if BASEDATE is not in the format MMDDYY.

The NUMDIGITS parameter is used only when the AGING report overflows the print line. Because of the number of ranges that AGING permits (8), and the variability of the length of the optional fields idfield1 and idfield2, it is possible for AGING to format a print line that overflows the linesize. The following steps, or a combination of steps, can be used to adjust the report to fit in the linesize:

- Eliminate the specification of one or both of idfield1 and idfield2.
- Decrease the number of ranges or eliminate the current range.
- Specify a NUMDIGITS which will create enough room for the line to print.

The default value for NUMDIGITS is 11. This allows detail items of up to 999 million. With the typical sizes of the amount, idfield1 and idfield2 fields, this allows approximately five or six intervals to be printed on a 132character linesize.

If the line overflows, and you want to reduce the number of digits printed in each interval, it is important to evaluate the possibility of truncating results. Each reduction of three digits specified for NUMDIGITS reduces four print positions per range (because of the comma).

You can calculate the amount by which NUMDIGITS must be reduced by dividing the overflow amount printed in diagnostic message B061 by the number of ranges specified (including the current range). For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*. Round this result upward to the nearest integer value and you have the number of print positions per range that must be eliminated.

When you use a REPORTTYPE of SUMMARY, the field idfield2 is not printed on the report. If you specify a value for idfield2, the report contains the appropriate number of blank columns.

Operation — Database

AGING can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be a nondatabase file name or the name of a database file defined in the library section.

Examples

The following are two examples of AGING.

Example One

This example demonstrates a standard AGING using only the mandatory parameters plus CURANGE and the idfields for informational purposes. The keyword parameters default to:

```
BASEDATE  = current system date
RANGE     = 30 days
INTERVALS = 3
REPORTTYPE = DETAIL
```

Input

```
FILE CUSTFIL FB (44 4400)
  DATE     1 6 N MASK 'Z9/99/99'
  CUSTNO   7 6 N
  INVNO   13 6 N
  BALANCE 19 6 P 2
%AGING1 CUSTFIL DATE MMDDYY BALANCE
%AGING2 CUSTNO INVNO CURANGE
```

Output

CA-PANAUDIT PLUS AGING REPORT

FILE NAME: CUSTFIL   DATE FIELD: DATE   AMOUNT FIELD: BALANCE

RANGE 1   RANGE 2   RANGE 3

3160   6190   OVER 90
DAYS OLD   DAYS OLD   DAYS OLD

AGING OF ACCOUNTS AS OF 3/12/90

DATE	CUSTNO	INVNO	CURRENT	RANGE 1	RANGE 2	RANGE 3	TOTAL
10/13/89	000001	000112	.00	.00	.00	5,830.00	5,830.00
11/08/89		000120	.00	.00	.00	5,800.00	5,800.00
12/21/89		000100	.00	.00	2,851.00	.00	2,851.00
12/23/89		000104	.00	.00	1,572.00	.00	1,572.00
12/25/89		000128	.00	.00	6,954.00	.00	6,954.00
1/31/90		000108	.00	1,070.00	.00	.00	1,070.00
2/03/90		000124	.00	7,650.00	.00	.00	7,650.00
2/16/90		000116	8,462.00	.00	.00	.00	8,462.00
	000001		8,462.00	8,720.00	11,377.00	11,630.00	40,189.00
9/05/89	000002	000109	.00	.00	.00	5,200.00	5,200.00
9/15/89		000105	.00	.00	.00	9,063.00	9,063.00
10/11/89		000121	.00	.00	.00	7,388.00	7,388.00
10/13/89		000101	.00	.00	.00	8,674.00	8,674.00
11/12/89		000125	.00	.00	.00	3,039.00	3,039.00
11/30/89		000117	.00	.00	.00	4,741.00	4,741.00
1/07/90		000113	.00	.00	2,745.00	.00	2,745.00
1/16/90		000129	.00	2,997.00	.00	.00	2,997.00
	000002		.00	2,997.00	2,745.00	38,105.00	43,847.00
10/24/89	000003	000122	.00	.00	.00	8,682.00	8,682.00
11/18/89		000106	.00	.00	.00	7,993.00	7,993.00
11/28/89		000110	.00	.00	.00	3,077.00	3,077.00
12/01/89		000126	.00	.00	.00	8,252.00	8,252.00
12/06/89		000102	.00	.00	.00	4,501.00	4,501.00
1/07/90		000118	.00	.00	8,216.00	.00	8,216.00
2/15/90		000114	9,589.00	.00	.00	.00	9,589.00
	000003		9,589.00	.00	8,216.00	32,505.00	50,310.00
9/20/89	000004	000119	.00	.00	.00	1,974.00	1,974.00
10/06/89		000123	.00	.00	.00	1,959.00	1,959.00
10/24/89		000111	.00	.00	.00	3,955.00	3,955.00
11/03/89		000115	.00	.00	.00	6,985.00	6,985.00
1/22/90		000103	.00	6,437.00	.00	.00	6,437.00
2/16/90		000127	9,726.00	.00	.00	.00	9,726.00
2/22/90		000107	9,815.00	.00	.00	.00	9,815.00
	000004		19,541.00	6,437.00	.00	14,873.00	40,851.00
			37,592.00	18,154.00	22,338.00	97,113.00	175,197.00

NUMBER OF RECORDS SELECTED FOR THIS REPORT: 30

Example Two

In this example, CURANGE is not specified, the summary footnotes about current records are included, and variable ranges are requested.

Input

```
FILE CUSTFIL FB (44 4400)
  DATE     1 6 N MASK 'Z9/99/99'
  CUSTNO   7 6 N
  INVNO   13 6 N
  BALANCE 19 6 P 2
%AGING1 CUSTFIL DATE MMDDYY BALANCE BASEDATE 030190 BASEFORM MMDDYY +
   NUMDIGITS 8 RANGE 60 RANGE1 30 RANGE2 30 RANGE3 40
%AGING2 CUSTNO INVNO INTERVALS 4
```

Output

```
                                   CA-PANAUDIT PLUS AGING REPORT

                     FILE NAME: CUSTFIL    DATE FIELD: DATE    AMOUNT FIELD: BALANCE

                            RANGE 1       RANGE 2       RANGE 3       RANGE 4

                            61  -  90     91  - 120     121  - 160    OVER    160
                            DAYS OLD      DAYS OLD      DAYS OLD      DAYS OLD

                                   AGING OF ACCOUNTS AS OF    3/01/90


  DATE    CUSTNO  INVNO      RANGE 1       RANGE 2       RANGE 3       RANGE 4             TOTAL

10/13/89  000001  000112          .00           .00      5,830.00           .00          5,830.00
11/08/89          000120          .00      5,800.00           .00           .00          5,800.00
12/21/89          000100     2,851.00           .00           .00           .00          2,851.00
12/23/89          000104     1,572.00           .00           .00           .00          1,572.00
12/25/89          000128     6,954.00           .00           .00           .00          6,954.00
          000001            11,377.00      5,800.00      5,830.00           .00         23,007.00

 9/05/89  000002  000109          .00           .00           .00      5,200.00          5,200.00
 9/15/89          000105          .00           .00           .00      9,063.00          9,063.00
10/11/89          000121          .00           .00      7,388.00           .00          7,388.00
10/13/89          000101          .00           .00      8,674.00           .00          8,674.00
11/12/89          000125          .00      3,039.00           .00           .00          3,039.00
11/30/89          000117          .00      4,741.00           .00           .00          4,741.00
          000002                  .00      7,780.00     16,062.00     14,263.00         8,105.00

10/24/89  000003  000122          .00           .00      8,682.00           .00          8,682.00
11/18/89          000106          .00      7,993.00           .00           .00          7,993.00
11/28/89          000110          .00      3,077.00           .00           .00          3,077.00
12/01/89          000126     8,252.00           .00           .00           .00          8,252.00
12/06/89          000102     4,501.00           .00           .00           .00          4,501.00
          000003            12,753.00     11,070.00      8,682.00           .00         32,505.00

 9/20/89  000004  000119          .00           .00           .00      1,974.00          1,974.00
10/06/89          000123          .00           .00      1,959.00           .00          1,959.00
10/24/89          000111          .00           .00      3,955.00           .00          3,955.00
11/03/89          000115          .00      6,985.00           .00           .00          6,985.00
          000004                  .00      6,985.00      5,914.00      1,974.00         14,873.00

                            24,130.00     31,635.00     36,488.00     16,237.00        108,490.00

NUMBER OF AGED RECORDS:                                                            20
NUMBER OF CURRENT RECORDS:                                                         10
NUMBER OF CURRENT AND AGED RECORDS:                                                30
```

```
TOTAL AMOUNT OF CURRENT RECORDS:                                          66,707.00
TOTAL AMOUNT OF CURRENT AND AGED RECORDS:                                175,197.00
```

ALPHACON

The ALPHACON routine converts an edited numeric field (alphanumeric) into a numeric field, which can be used for calculations. If the input field contains alphabetic characters, they are removed by ALPHACON in the conversion. The number of decimal places in the converted number is defined in the file definition.

CA-Easytrieve Plus edit masks allow the use of any character as a negative number indicator, but ALPHACON only checks the most typical negative number indicators:

- Minus sign (-)
- Credit indicator (CR)
- Two debit indicators (DB, DR)

If one of these negative indicators is found, ALPHACON converts the indicator to a minus sign (-) and places it at the end of the converted number. If ALPHACON encounters a negative indicator that it does not recognize, ALPHACON ignores the indicator, and the number is converted to a positive number.

Syntax

```
%ALPHACON    alphafield  numfield     [DECIMAL 'separator']
```

`alphafield`

Specify the alphanumeric field to be converted. The field has a maximum length of 30 characters.

`numfield`

Specify the numeric field to which the converted number is written. The converted number is truncated to the number of places specified in the field definition. The maximum number of digits supported is 17.

`[DECIMAL 'separator']`

Specify the symbol used as a separator between the integer and the decimal in the input field if it is not a decimal point (**.**).

Operation — Inline

ALPHACON generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in specification of parameters is required to use ALPHACON with database files.

Input

In this example, ALPHACON converts the field into numeric data. ALPHACON-FLAG is set to YES if the value is converted correctly. If the field is truncated on the right, left, or on both sides, ALPHACON-FLAG is set to RIGHT, LEFT, or BOTH, respectively.

```
FILE INFILE CARD
ALPHA      1  30  A
NUMBER     W  17  N  5
*
JOB INPUT INFILE
%ALPHACON ALPHA NUMBER
PRINT ALPHACON-REPORT
REPORT ALPHACON-REPORT LINESIZE 78
TITLE 1 'ALPHANUMERIC CONVERSION'
LINE ALPHA ALPHACON-FLAG NUMBER
END
```

Output

```
5/20/88                   ALPHANUMERIC CONVERSION             PAGE      1

            ALPHA                 ALPHACON-FLAG           NUMBER

 1                                    YES                        1.00000
 -2.3                                 YES                        2.30000-
 456,789.0123456789012               RIGHT                   456,789.01234
 $****3,157.36                        YES                      3,157.36000
 5711.837392-                        RIGHT                     5,711.83739-
 38,351.99CR                          YES                     38,351.99000-
 1,398,351.23DR                       YES                  1,398,351.23000-
 123.45DB                             YES                        123.45000-
 1#800#111#2222                       YES             18,001,112,222.00000
 A1B2C3D4E5F6G7H8I9J0K1L2M3N          LEFT           234,567,890,123.00000
```

ALPHAGEN

The ALPHAGEN routine generates alphanumeric data and writes it to the field parameter.

Note: The FILEGEN routine must be invoked to use the ALPHAGEN routine.

Syntax

```
                            {BETWEEN minimum maximum             }
%ALPHAGEN field 'literal'   {SEQUENCE from to increment          }
                            {CONSTANT 'value1,value2, ... ,valuen'}
```

`field`

Specify the name of the field where ALPHAGEN places the alphanumeric data it generates. This field must be large enough to contain the characters specified by the literal parameter plus the characters specified by the BETWEEN, SEQUENCE, or CONSTANT parameters. Valid names include any alphanumeric field defined in the file name specified in FILEGEN. The field must have a data format of A (alphanumeric).

`'literal'`

Specify any combination of alphanumeric characters to be used as a prefix for data generated by the BETWEEN, SEQUENCE, or CONSTANT parameters. Valid literals include any sequence of 1 through 15 alphanumeric characters. Use single quotation marks around the data.

`{BETWEEN minimum maximum}`

The BETWEEN keyword causes random numbers to be generated in the range specified by the minimum and maximum parameters that you specify. The value is appended to the literal parameter and written to the field parameter. Valid values for minimum and maximum are either actual numeric values or the name of a field containing the value. Minimum and maximum cannot be more than 15 digits. The seed for the random generation of values is obtained from the FILEGEN routine.

Values generated are greater than or equal to the minimum and less than the maximum.

`{SEQUENCE from to increment}`

The SEQUENCE keyword causes a fixed set of numbers to be generated. The set of numbers begins with the from value and is incremented for each record by the increment value until the to value is equaled or exceeded. The sequence is repeated beginning with the from value. Each number is appended to the literal parameter and written to the field parameter. Valid values for from, to, and increment are actual numeric values or the name of a field containing the value.

To generate a decreasing set of numbers, specify a from value greater than the to value, and code a negative increment.

```
{CONSTANT 'value1, value2,. . .,valuen'}
```

The CONSTANT keyword generates a specified series of values. The sequence is repeated until a value has been generated for each record being created. Each number is appended to the literal parameter and written to the field parameter.

Separate the values in the constant string by commas and enclose the string in single quotation marks. Valid values for value1 to valuen consist of actual alphanumeric values 1 through 15 characters in length. The entire length of the literal portion of CONSTANT is limited to 40 characters, including commas.

Operation — Inline

Use the ALPHAGEN routine only after you have specified the FILEGEN routine. ALPHAGEN can be specified with DATEGEN, NUMGEN, and BADGEN. Conditional execution of ALPHAGEN is discussed in Conditional Execution of Data Generation Routines following the BADGEN routine.

Operation — Database

ALPHAGEN, with FILEGEN, cannot be used in a database application.

Examples

The following example illustrates how ALPHAGEN is used with FILEGEN:

```
FILE CENTFILE F(24)
  NAME    1 12 A
  EMP#   13  5 N
  BIRTH  18  6 N
*
JOB INPUT NULL
  %FILEGEN CENTFILE 25 5 NOHEX
  %ALPHAGEN NAME ‘ CUST ‘ CONSTANT ‘JOHNSON,SMITH,PETERS’
  %NUMGEN EMP# SEQUENCE 1 10050 1
  %DATEGEN BIRTH MMDDYY 0  BETWEEN 010151 010175
```

The following examples illustrate the three uses of ALPHAGEN. An example of the full facilities of the test data generation routines is found following the BADGEN routine.

Between

This example generates a random value between 10 and 94. Each value is appended to the literal PERSON and written to the NAME field.

```
%ALPHAGEN NAME 'PERSON' BETWEEN 10 95
```

Sequence

A value of 4 is appended to the literal PERSON and written to the NAME field of the first record, PERSON6 is written to the second record, PERSON8 to the third record, and so forth, in increments of 2 until the to value (98) is equaled or exceeded. The sequence is repeated starting with PERSON4.

```
%ALPHAGEN NAME 'PERSON' SEQUENCE 4 98 2
```

Constant

J B JOHNSON is written to the NAME field of the first record, J B SMITH to the second record, and J B PETERS to the third record. This sequence is repeated until the NAME field has been generated for each record being created.

```
%ALPHAGEN NAME 'J B' CONSTANT  ' JOHNSON, SMITH, PETERS'
```

APR

The APR routine calculates the effective annual percentage rate for a given annual percentage rate and period. The formula for calculating the effective annual percentage rate is:

```
EFFAPR = ((1 + apr)/period) ** poc - 1
```

Where:

apr—annual percentage rate

period—The number of times interest is compounded in a year. For example, if the interest rate is compounded quarterly, period = 4; if monthly, period = 12; if daily, period = 365.

poc—Number of periods compounded in a year. For example, to compound yearly, and the period = 12, then poc = 12. To compound quarterly, and period = 12, poc = 3.

To compute periods of continuous compounding, set period and poc to the same value.

Syntax

```
%APR    apr    period    poc    effapr
```

apr

Specify the annual percentage rate to use. A valid value is an actual numeric value or the name of a field containing a numeric value.

`period`

Specify the number of times in a year the interest is compounded. A valid value is an actual numeric value or the name of a field containing a numeric value. The value of period cannot be 0. If the value of period is 0, diagnostics message PAP310 is returned, and APR terminates. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*.

`poc`

Specify the number of periods of compounding in a year to use in calculating the effective annual percentage rate. A valid value is an actual numeric value or the name of a field containing a numeric value.

`effapr`

Specify the name of the numeric field to write the calculated effective annual percentage rate.

Operation — Inline

APR generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in specification of parameters is required to use APR with database files.

Example

In this example, the effective APR is calculated and printed.

Input

```
FILE INFILE
PERCRATE     1 2 N  HEADING ('PERCENTAGE' 'RATE')
PERD         3 2 N HEADING ('NUMBER' 'OF' 'PERIODS')
POC          5 2 N HEADING ('PERIODS' 'OF' 'COMPOUND')
*
DEFINE EFFAPR W 10 N 2 HEADING ('ANNUAL' 'PERCENTAGE' 'RATE')
*
JOB INPUT INFILE
  %APR PERCRATE PERD POC  EFFAPR
  PRINT APRDATA
*
REPORT APRDATA LINESIZE 78
 TITLE  'APR TEST DATA EXAMPLES'
 LINE PERCRATE PERD POC EFFAPR
```

Output

```
4/13/88                 APR TEST DATA EXAMPLES         PAGE      1

                            NUMBER    PERIODS       ANNUAL
              PERCENTAGE      OF        OF        PERCENTAGE
                 RATE      PERIODS   COMPOUND        RATE

                  01          01        12          12.68
                  02          02        12          12.68
                  03          03        12          12.68
                  04          14        12           3.48
                  05          05        12          12.68
                  06          01        12         101.21
                  07          12        12           7.22
                  07          12        24          14.98
                  07          12        01            .58
                  08          23        12           4.25
                  09          04        12          30.60
                  10          14        12           8.91
                  11          25        22          10.14
                  12          06        12          26.82
                  13          17        12           9.57
                  14          08        12          23.14
                  15          09        12          21.93
                  16          10        12          20.98
                  17          11        12          20.20
                  18          12        12          19.56
```

ATTPCT

The ATTPCT routine calculates the percentage (%) of a file's total records that constitutes a representative sample. The calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Expected error rate

A report lists the input parameters and the calculated sample percentage.

Use the ATTSAMP routine to calculate the appropriate sample size and randomly select the records from a file.

Syntax

```
%ATTPCT size confidence precision error
```

`size`

Specify the total number of records in the population being examined. A valid value is either an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ by more than the specified precision from the result that is obtained by examining the entire population.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. This parameter can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`precision`

Specify a range of tolerance, expressed as a percentage, that indicates the acceptable variance plus (+) or minus (-) in the calculations.

For example, if you estimate the occurrence of an attribute in a given population is 10 percent with a precision of 3.75 percent, this means you anticipate that the occurrence can be as low as 6.25 percent or as high as 13.75 percent. The true answer concerning the population's attribute can fall in this range at the specified confidence level. The precision parameter can be specified as an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`error`

Specify the percentage of error you estimate will be found as a result of the test. The estimated error rate can be guided by the results of a previous test, a preliminary survey, or a small pilot test of transactions. The error parameter can be specified as an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

Operation — Inline

ATTPCT lets you study the result of attribute sampling with a particular set of parameters without reading an input file. This provides a technique for studying the results so that you can make the best possible parameter selection for the given application. ATTPCT can be invoked any number of times, letting you study the effects of varying the parameters. See the example on the following page.

For example, you can use it to evaluate the effects on the sample size of varying the precision percentage.

When satisfactory results have been obtained, you can select the proper parameter and run the ATTSAMP routine (which produces the identical results), to randomly select the desired samples from a file.

If ATTPCT is used with a JOB INPUT NULL statement, the job must contain a STOP statement.

Note: The value for precision is specified as a percentage. (Precision in the VARPCT and VARSAMP routines, described later in this guide, is expressed as an absolute amount.)

Operation — Database

No change in the specification of parameters is required to use ATTPCT with database files.

Example

The following is an example of ATTPCT.

Input

```
JOB INPUT NULL
...
%ATTPCT 100000 90 1.8 3.2
%ATTPCT 100000 95 1.8 3.2
%ATTPCT 100000 98 1.8 3.2
...
STOP
```

Output

```
                    ATTRIBUTES SAMPLING REPORT

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT

        100,000        90           1.80         3.20     0.26000000%

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT

        100,000        95           1.80         3.20     0.36600000%

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT

        100,000        98           1.80         3.20     0.51600000%
```

This example demonstrates a technique for evaluating the effects on the sample percentage by varying the confidence factor. The input consists of three invocations of ATTPCT with identical values for population size, precision, and error rate, but varying confidence levels of 90, 95, and 98 percent. As the requested confidence level for the sample increases, the required sample percentage value increases. The report lists the input values and the different sample percents produced for the values specified.

ATTSAMP

The ATTSAMP routine calculates the percentage (%) of a file's total records that constitutes a representative sample and selects the appropriate number of records randomly from the file.

The calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Expected error rate

Selected records can be written to a sample file. A report lists the input parameters and the result of the sample size calculation.

Use the ATTPCT routine to calculate the appropriate sample size without selecting records.

Syntax

```
%ATTSAMP1 infile size confidence precision error seed

%ATTSAMP2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to ATTSAMP. A valid name is any previously defined file.

`size`

Specify the total number of records in the population being examined. A valid value is either an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ by more than the specified precision from the result that is obtained by examining the entire population.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances it is not representative. The confidence percentage specified must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. This parameter can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`precision`

Specify a range of tolerance, expressed as a percentage, that indicates the acceptable variance plus (+) or minus (-) in the calculations.

For example, if you estimate the occurrence of an attribute in a given population is 10 percent with a precision of 3.75 percent, this means you anticipate that the occurrence can be as low as 6.25 percent or as high as 13.75 percent. The true answer concerning the population's attribute can fall in this range at the specified confidence level. The precision parameter can be specified as an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`error`

Specify the percentage of error that you estimate will be found as a result of the test. This estimated error rate can be guided by the results of a previous test, a preliminary survey, or a small pilot test of transactions. The error parameter can be specified as an actual numeric value or the name of a field containing a numeric value. Values may contain up to two decimal places and are truncated on the right if more than two are specified.

`seed`

Specify an arbitrary number that initiates the internal random number generator. This seed is used to randomize the selection of samples from the file. A valid value is either an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are to be written to an output file.

outfile—records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

`[DBFILE infile]`

This optional parameter specifies a database file for use with ATTSAMP. Infile identifies the name of the input file to ATTSAMP. The name must be the same name that you specified for infile on the first invocation statement.

`[PERFORM procname]`

Specify the name of a CA-Easytrieve Plus procedure that is performed by the ATTSAMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field ATTSAMP-SELECTED is set to the value YES. If a record is not selected for the sample file, ATTSAMP-SELECTED is set to the value NO.

After the invocation of ATTSAMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the ATTSAMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that ATTSAMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You can adjust the size parameter when screening code is inserted that causes records to be bypassed from ATTSAMP processing. The value specified for the size parameter must represent the size of the population being examined for the attribute sampling. For the resulting sample percentage and optional sample file to be accurate, the size parameter must be adjusted by the number of records that are bypassed.

For example, if you specify 50,000 as the file size, and screening code causes 25,000 of these records to be bypassed, the population size sampled by ATTSAMP is actually 25,000. The calculated percentage is therefore incorrect, and the sample file created is also invalid.

To avoid this result, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent a sample file from being created. Notice the number of records processed by ATTSAMP in the report. You can rerun the job using the record count listed in the report for the size parameter and specify an output file name in place of NOFILE. This ensures the correct results from ATTSAMP processing while bypassing the unwanted records.

Note: The value for precision is specified as a percentage. (Precision in the VARPCT and VARSAMP routines, described later in this chapter, is expressed as an absolute amount.)

Operation — Database

The DBFILE parameter identifies ATTSAMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using ATTSAMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of ATTSAMP.

Example

Input

```
FILE INFILE FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE FB (44 4400)
%ATTSAMP1 INFILE 10000 90 2.5 4.0 928451
%ATTSAMP2 OUTFILE
```

Output

```
                  ATTRIBUTE SAMPLING REPORT

                      INPUT PARAMETERS

        INPUT FILENAME                         INFILE
        TOTAL POPULATION SIZE                  10,000
        REQUIRED PRECISION                          2.50
        REQUIRED CONFIDENCE LEVEL                 90
        ERROR RATE                                  4.00

                      SAMPLE RESULTS

        SAMPLE PERCENTAGE REQUIRED                  1.65000000%
        SAMPLE SIZE REQUIRED                     165

                      SAMPLE FILE

        NUMBER OF RECORDS PROCESSED            10,000
        NUMBER OF RECORDS REQUESTED               165
        NUMBER OF RECORDS IN SAMPLE FILE          165

        FILE OUTFILE WILL BE CREATED
```

The report lists the input parameters followed by the results of the ATTSAMP calculations. The report also provides the results of the random sampling process, including whether an output file is created.

BADGEN

The BADGEN routine generates invalid data by overlaying data previously created by one of the data generation routines. You must invoke the appropriate routine (ALPHAGEN, DATEGEN, or NUMGEN) before invoking BADGEN, so that a particular field can be overlaid with invalid data.

For numeric fields, BADGEN generates non-numeric data; for alphanumeric fields, BADGEN generates nonalphanumeric characters. Invalid data is generated randomly for the specified percentage of the file.

Syntax

```
%BADGEN field percent [RSGROUP number]
```

`field`

Specify the name of the field that you want BADGEN to overlay with invalid data. The name must be the same as the field parameter specified on a previous data generation routine.

`percent`

Specify a numeric value between 1 and 99 that represents a percent of all records being created (as established by the FILEGEN routine). This parameter represents the percent of records that will be generated with invalid data.

`[RSGROUP number]`

This optional parameter specifies a numeric value that starts at 1 and is incremented by 1 for each BADGEN coded in a FILEGEN. The default value is 1. If only one BADGEN is used, this parameter can be omitted.

Operation — Inline

You must invoke BADGEN after the datageneration routine for which it is creating invalid data.

Operation — Database

BADGEN, used with FILEGEN, ALPHAGEN, DATEGEN, and NUMGEN, cannot be used in a database application.

Example

Input

```
FILE ...
  ZONE      1   5   N
  ...
%NUMGEN ZONE BETWEEN 3 98
%BADGEN ZONE 15
```

In this example, NUMGEN generates a random value between 3 and 97 for the ZONE field. BADGEN overlays 15 percent of these values with non-numeric data.

Generating Calculated Test Data

In addition to using NUMGEN, you can generate numeric test data by means of a calculation. Additionally, the calculation can be preceded by conditional logic that states the conditions under which the statements or statements are executed. The following example demonstrates data generated with conditional logic and calculations.

Example

```
FILE PAYFILE ...
  GROSS-PAY ...
  TAX-CLASS ...
  DEDUCTIONS ...
```

```
  BIRTH ...
JOB INPUT NULL
%FILEGEN PAYFILE 200 13243 HEX
%NUMGEN GROSS-PAY BETWEEN 200 1000
%NUMGEN TAX-CLASS CONSTANT '2,3,1,3,1,2'
IF TAX-CLASS   =  1
   DEDUCTIONS  =  GROSS-PAY  *  .1
END-IF
IF TAX-CLASS   =  2
   DEDUCTIONS  =  GROSS-PAY  *  .15
END-IF
IF TAX-CLASS   =  3
   DEDUCTIONS  =  GROSS-PAY  *  .18
END-IF
%DATEGEN BIRTH MMDDYY BETWEEN 010130 123165
```

This example contains a series of formulas that calculate the value for DEDUCTIONS. Each formula is preceded by an IF statement containing the TAX-CLASS field generated by NUMGEN to determine which formula is used.

Note: A value is generated for both the GROSS-PAY and TAX-CLASS fields prior to the logic that determines the value for DEDUCTIONS.

Conditional Execution of Data Generation Routines

Any data generation routine can be conditionally executed by preceding the invocation of the routine with an IF statement and following it with an END-IF statement. This allows data to be generated for a field based on the decision logic. You can have multiple invocations of the data generation routines for the same field.

To facilitate the conditional generation of data, the FILEGEN routine maintains two fields:

FILECOUNT—The number of the record currently being created.

USER-RANDOM—Random number between 0.0 and 99.9.

The following example demonstrates the conditional execution of data generation routines.

Example

```
FILE PAYFILE ...
  GROSS-PAY ...
  TAX-CLASS ...
  BIRTH ...
JOB INPUT NULL
%FILEGEN PAYFILE 200 13243 HEX
%NUMGEN TAX-CLASS CONSTANT '2,3,1,3,1,2'
IF FILE-COUNT LE 150
   %NUMGEN GROSS-PAY BETWEEN 200 500
END-IF
IF FILE-COUNT GT 150
   %NUMGEN GROSS-PAY SEQUENCE 500 1000 50
END-IF
%DATEGEN BIRTH MMDDYY BETWEEN 010130 123165
```

In this example, the NUMGEN routine is coded twice for the GROSS-PAY field. For records 1 through 150, the NUMGEN routine generates a random GROSS-PAY value of between 200 and 499. For records 151 to 200, NUMGEN generates a fixed set of numbers beginning with 500, and incrementing the value by 50 until 1000 is reached. This sequence is repeated until a GROSS-PAY value is generated for all 200 records.

Generating Invalid Data

You can generate invalid data for a field by one of two methods:

- Use BADGEN to overlay a field with bad data.
- Code a routine to generate invalid data. This method is demonstrated in the following example.

Example

```
FILE PAYFILE ...
  REGION ...
  ZONE ...
JOB INPUT NULL
%FILEGEN PAYFILE 200 13243 HEX
IF USER-RANDOM  LT  90
   %NUMGEN REGION BETWEEN 1 10
ELSE
   %NUMGEN REGION BETWEEN 10 101
END-IF
IF USER-RANDOM  LT  30
   %NUMGEN ZONE CONSTANT '1,2,3'
END-IF
IF USER-RANDOM  EQ  30 THRU 70
   %NUMGEN ZONE CONSTANT '4,5,6,7,8'
END-IF
IF USER-RANDOM  GT  70
   %NUMGEN ZONE CONSTANT '12,15,17'
END-IF
```

Assume that the valid values for REGION are the numbers 1 through 9. This example generates valid values in approximately 90 percent of the records. The remaining records contain invalid values between 10 and 100.

This example also illustrates how to use USER-RANDOM to control the approximate distribution of values. Assuming that valid values for ZONE are the numbers 1 through 8, the conditional logic generates valid ZONEs for approximately 70 percent of the records. In addition, approximately 30 percent contain the values 1, 2, or 3, and 40 percent contain 4, 5, 6, 7, or 8. The remaining records contain the invalid values 12, 15, or 17.

CAVEVAL

The CAVEVAL routine performs an evaluation of the sample that CAVSAMP creates. Audited amounts for all recorded amounts in the CAVSAMP sample file must be established before CAVEVAL can perform the evaluation.

CAVSAMP and CAVEVAL are valid only for audit applications involving overstatements because CAVEVAL bases the evaluation only on overstatements of the recorded amounts. Depending on the number and the amount of overstatements of recorded amounts over audited amounts, CAVEVAL calculates an upper limit of error. If this upper limit is less than the tolerable error specified for both CAVSAMP and CAVEVAL, a positive conclusion can be made.

The positive conclusion is that the sample results support the conclusion that, at the specified confidence, the population is not overstated by more than the tolerable error. If the upper limit is greater than the tolerable error, the sample results do not support the conclusion.

Due to the conservative nature and the one-sided approach of this method, when the upper limit of error is greater than the tolerable error, it is possible that no material overstatement in the population exists. When the upper limit of error is greater than the tolerable error, there is no realistic estimate of the actual maximum amount of overstatement. Furthermore, statistics regarding understatements are given in the CAVEVAL report for informational purposes and do not participate in any of the statistical calculations.

The sample file that CAVSAMP produces must be analyzed only by the CAVEVAL evaluation routine. Any other use of the sample that CAVSAMP creates does not follow accepted auditing procedures. Due to the mathematical properties of the algorithm that CAVEVAL uses, audited amounts that are less than zero may cause an exaggeration of the upper error limit. Also, due to the impact this may have on the conclusion that the evaluation routine makes, it is a common audit procedure to separately examine audited amounts that are less than zero.

CAVEVAL notices this condition by providing a warning message. For additional details regarding this condition, see Operation — Stand-alone DISPLAY in this routine.

Syntax

```
%CAVEVAL1 infile recamt audamt tolerror experror confidence
```

```
%CAVEVAL2
```

`infile`

Specify the name of the input file to CAVEVAL. This must be the sample file that the CAVSAMP routine creates. A valid name is any previously defined file.

`recamt`

Specify the name of the quantitative field containing the recorded amount for each record. A valid name is any quantitative field defined in the input file.

`audamt`

Specify the name of the quantitative field containing the audited amount for each record. A valid name is any quantitative field defined in the input file.

`tolerror`

Specify the tolerable amount of error for the total population. The tolerable error will be compared to the upper limit of error that CAVEVAL calculates. The value for tolerror must be the same as the value you specified for tolerror in the CAVSAMP routine. A valid value is an actual numeric value or the name of a field containing a numeric value greater than zero.

`experror`

Specify an expected amount of error for the total population. The value of experror must be the same as the value you specified for experror in the CAVSAMP routine. A valid value is an actual numeric value or the name of a field containing a numeric value greater than zero.

`confidence`

Specify a numeric value that represents the confidence percentage. The probability of the confidence percentage is that the CAVEVAL routine is correct. The positive conclusion of the CAVEVAL routine is that the population is not overstated by more than the tolerable error. The confidence percentage must be one of the following: 99, 97.5, 95 90, 85, 80, 75, 66, or 50. The value must be the same value that you specified for confidence in the CAVSAMP routine. You can specify this parameter as an actual numeric value or the name of a field containing a numeric value.

Operation — Stand-alone DISPLAY

CAVEVAL calculates the overstatements from the input file. An overstatement exists when the recorded amount is greater than the audited amount. For each overstatement, a tainting value is calculated. The tainting value is the ratio of the amount of overstatement to the recorded amount:

```
(recamt - audamt) / absolute value(recamt)
```

The tainting value is a component in the calculation of a value for the projected error for this overstatement. The tainting value is typically between 0 and 1.0 and is formulated to perform similar to a percentage.

However, when the audited amount for an overstatement becomes less than zero, the tainting value can increase rapidly. For example, if the recorded amount is 10.00, and the audited amount is -100.00, the tainting is 11, or 1100 percent. This is equivalent to one overstatement in the sample file creating eleven 100 percent overstatements.

If this sample result is allowed to participate in the CAVEVAL evaluation, the upper limit of error may be grossly overstated. In many applications, this is considered to be an exaggeration of the projected error for this record. This is why audited amounts of less than zero are commonly eliminated from the sample file and audited with separate procedures.

When audited amounts for overstatements that are less than zero are detected, CAVEVAL issues a warning message that prints the number of the tainting values and the largest tainting value. The tainting value that prints is a percentage. When the number of sample records with overstated audited amounts less than zero becomes high in relation to the total number of sample items and/or the largest tainting value becomes large (about 500 percent), this may cause an exaggerated upper limit of error. If you suspect that the upper limit of error for your result is high, create a listing of all items in the sample file with overstated audited amounts less than zero.

The projected errors for all overstatements are components in an equation that calculates the upper limit of error. The upper limit of error is the basis for a positive or negative conclusion regarding the accuracy of the population based on the recorded and audited amounts in the sample file.

Operation — Database

CAVEVAL cannot be used in a database application.

Example

The following is an example of CAVEVAL.

Input

```
FILE SAMPLE FB (44 4400)
RECAMT    1   10  N  2
AUDAMT    11  10  N  2
%CAVEVAL1 SAMPLE RECAMT AUDAMT 150000 30000 90
%CAVEVAL2
```

Output

```
                    CAVEVAL EVALUATION REPORT

                        INPUT PARAMETERS

          INPUT SAMPLE FILE                                   SAMPLE
          RECORDED AMOUNT FIELD                               RECAMT
          AUDITED AMOUNT FIELD                                AUDAMT
          TOLERABLE ERROR                                 150,000.00
          EXPECTED ERROR                                   30,000.00
          REQUIRED CONFIDENCE LEVEL                            90.00


                        EVALUATION REPORT

          TARGET VALUE                                     45,454.54
          NUMBER OF RECORDS IN SAMPLE FILE                    113

          NUMBER OF UNDERSTATEMENTS                             6
          TOTAL AMOUNT OF UNDERSTATEMENTS                   7,486.75-

          NUMBER OF OVERSTATEMENTS                              5
          TOTAL AMOUNT OF OVERSTATEMENTS                    3,607.15

          ACTUAL ERROR                                      3,607.15
          TOTAL PROJECTED ERROR                            22,727.25
          TOTAL ALLOWANCE FOR SAMPLING RISK               113,954.51
          UPPER LIMIT OF ERROR                            136,681.76

                           CONCLUSION

THE UPPER LIMIT IS LESS THAN THE TOLERABLE ERROR. THEREFORE,
THE SAMPLE RESULTS SUPPORT THE CONCLUSION THAT, AT THE SPECIFIED CONFIDENCE,
THE POPULATION IS NOT OVERSTATED BY MORE THAN THE TOLERABLE ERROR.

                        EVALUATION NOTES

THE COMBINED ATTRIBUTES AND VARIABLES PROPORTIONAL SAMPLING METHOD IS
A ONE-SIDED APPROACH AND IS LIMITED TO OVERSTATEMENTS ONLY.
STATISTICS ON UNDERSTATEMENTS ARE PRINTED FOR INFORMATIONAL PURPOSES ONLY.
UNDERSTATEMENTS DO NOT AFFECT THE CALCULATION OF THE UPPER LIMIT OF ERROR.
BECAUSE OF THIS, AND THE NATURE OF THE ALGORITHM EMPLOYED BY THIS METHOD,
RESULTS AND CONCLUSIONS ARE CONSERVATIVE IN NATURE.
```

First, the input parameters are listed. This is followed by the evaluation report that lists the target value, the number of records in the sample file, the number and amount of understatements and overstatements, and the components that determine the upper limit of error.

Understatements do not participate in the calculations that determine the upper limit of error and are presented for informational purposes only. The amount and frequency of overstatements have a direct relationship in calculating the value of the total projected error.

The actual error is the total amount of deviation found in all overstatements. The total projected error is a measurement of the amount of error projected on the population as a result of the overstatements found in the sample file. The total allowance for sampling risk is an estimated amount of error in the population based on the number of errors found, the size of the errors, and the specified confidence level. The projected error and allowance for sampling risk are added to produce a value for the upper limit of error.

If the upper limit of error is less than the tolerable error, a conclusion can be made that, at the specified confidence level, the population is not overstated by more than the tolerable error. If the upper limit of error is greater than or equal to the tolerable error, the conclusion is not supported, and the following message is printed:

```
THE UPPER LIMIT IS GREATER THAN OR EQUAL TO THE TOLERABLE ERROR, THEREFORE,
SAMPLE RESULTS DO NOT SUPPORT CONCLUSION THAT, AT THE SPECIFIED CONFIDENCE,
THE POPULATION IS NOT OVERSTATED BY MORE THAN THE TOLERABLE ERROR.
ADDITIONAL SAMPLING OR TESTING MAY BE NECESSARY.
```

In this case, you may require additional sampling or alternate sampling methods.

The evaluation notes serve as a reminder of the one-sided approach of this sampling and evaluation method and that the results are conservative in nature. This is explained in the description of the CAVEVAL routine.

CAVSAMP

The CAVSAMP routine creates a sample file based on a combined attributes and variables proportional sampling algorithm. The AICPA recommends this statistical method. This statistical method is presented in numerous publications that discuss proportional sampling techniques.

Specifically, CAVSAMP determines a target value based on the input values of tolerable error, expected error, and confidence. This target value is identical to the parameter required for SPS sampling. The target value is used to create an SPS sample, where the probability of selecting a record is proportional to the size of the value of that record. For a description of SPS, see the SPS routine.

CAVSAMP uses an upperlimit approach and is valid only for audit applications involving overstatements. CAVSAMP is valid only for use with large populations (at least 1000, and preferably 5000 records). Use the CAVEVAL evaluation routine to make conclusions regarding the sample created by the CAVSAMP routine. Any other use of the sample that this routine creates does not follow accepted auditing procedures. The description of the CAVEVAL routine covers the various limitations and interpretations of the sample results.

Syntax

```
%CAVSAMP1 infile field tolerror experror confidence seed

%CAVSAMP2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to CAVSAMP. A valid name is any previously defined file.

`field`

Specify the name of the field that is in the SPS sampling algorithm that the CAVSAMP routine uses. A valid name is any numeric field defined in the input file.

`tolerror`

Specify a tolerable amount of error for the total population for the field being audited in the input file. The tolerable error is the maximum amount of overstatement acceptable in a given audit application. A valid value is either an actual numeric value or the name of a field containing a numeric value greater than zero.

`experror`

Specify an expected amount of error for the total population for the field being audited in the input file. The expected error is the amount of error the auditor expects to find in the population. A valid value is an actual numeric value or the name of a field containing a numeric value greater than zero.

`confidence`

Specify a numeric value that represents the confidence percentage. The probability of the confidence percentage is that the CAVEVAL routine is correct. The positive conclusion of the CAVEVAL routine is that the population is not overstated by more than the tolerable error. The confidence percentage must be one of the following: 99, 97.5, 95, 90, 85, 80, 75, 66, or 50. You can specify this parameter as an actual numeric value or the name of a field containing a numeric value.

`seed`

Specify an arbitrary number that initiates the random number generator for the SPS algorithm that CAVSAMP uses. The seed initializes a work field that accumulates the input value. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length, with no decimal places. Values greater than seven digits are truncated on the left.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are written to an output file.

outfile–Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE–Records selected for the sample are not written to an output file.

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with CAVSAMP. Infile identifies the name of the input file to CAVSAMP. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure that is performed by the CAVSAMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field CAVSAMP-SELECTED is set to the value YES. If a record is not selected for the sample file, CAVSAMP-SELECTED is set to the value NO.

After the invocation of CAVSAMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the CAVSAMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that CAVSAMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system will substitute a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

CAVSAMP uses the tolerable error, expected error, and confidence parameters to establish a target value for an SPS sampling. After creation of the SPS sample, you can enter into all records audited amounts for all recorded amounts. Then input this file to the CAVEVAL routine. The CAVEVAL routine performs an evaluation of the difference of recorded and audited amounts and presents the appropriate conclusion.

The tolerable error parameter is the maximum amount of monetary error that can exist without causing the book value of the field being audited to be materially overstated. The expected error is the estimated amount of monetary error in the population. A degree of professional judgment is required to select values for these parameters. Information from previous audits, pilot samples, or other sources provides additional guidance in selecting values for these parameters.

You must evaluate the sample file that CAVSAMP produces only with the CAVEVAL routine. Any other evaluations you make from the sample file may produce incorrect conclusions. Also, this sampling and evaluation method uses an upperlimit approach, and you can apply this method only to audit situations involving overstatements.

For more information about the conclusions made through the use of this routine, see the description of the CAVEVAL routine.

Operation — Database

The DBFILE parameter identifies CAVSAMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using CAVSAMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of CAVSAMP.

Example

The following is an example of CAVSAMP.

Input

```
FILE INFILE    FB (44 4400)
RECAMT      1   10  N  2
FILE SAMPLE    FB (44 4400)
%CAVSAMP1 INFILE RECAMT 150000 30000 90 1357531
%CAVSAMP2 SAMPLE
```

Output

```
                CAVSAMP SAMPLING REPORT

                   INPUT PARAMETERS

INPUT FILENAME                       INFILE
INPUT FIELD                          RECAMT
TOLERABLE ERROR                    150,000.00
EXPECTED ERROR                      30,000.00
CONFIDENCE VALUE                        90.00


                    SAMPLE FILE
```

```
NUMBER OF RECORDS PROCESSED           1,000
TOTAL VALUE OF RECORDS PROCESSED   5,139,731.67
TARGET VALUE                          45,454.54
NUMBER OF RECORDS IN SAMPLE FILE        113

FILE SAMPLE WILL BE CREATED
```

In the sampling report, the input parameters are listed first, followed by the sample results. One thousand (1,000) records are processed from the input file with a total value of 5,139,731.67. Using the values listed for tolerable error, expected error, and confidence, a target value of 45,454.54 is calculated. This target value is used as input to an SPS algorithm that selects 113 records for the sample file. The last line of output indicates that the output file is created.

Audited amounts for the RECAMT field must now be obtained and entered into the sample file. After entering all audited amounts, the CAVEVAL routine calculates the upper limit of error and makes the appropriate conclusion about the population. The output is an example of CAVSAMP.

CONVAE

The CONVAE routine converts ASCII alphanumeric characters to their EBCDIC equivalent. CONVAE is an inline routine that you can use with other CA-PanAudit Plus Routines O-R CA-Easytrieve Plus logic. The inline design of CONVAE reduces execution time and reduces requirements for temporary or permanent storage space. You can also use CONVAE to create a permanent file of converted data.

For conversion from EBCDIC to ASCII, see the CONVEA routine.

Syntax

```
%CONVAE [DBFILE] [STARTPOS identifier] [LENGTH value]
```

`[DBFILE]`

Specify this optional parameter only for database use of CONVAE. Specify only the literal DBFILE, not the name of the active input file.

`[STARTPOS identifier]`

This optional parameter specifies the starting position for the conversion process. The identifier must be a previously defined field.

Conversion takes place in the active input file starting at the position defined by STARTPOS and continuing for the length specified by the length parameter.

The default value for STARTPOS is the first byte of the active input file. This means that when using the default value the conversion begins with the first byte of the active input file.

The requirements for specifying STARTPOS are different when you use CONVAE in a database application. In this case the STARTPOS parameter is no longer optional—it is a required parameter.

`[LENGTH value]`

This optional parameter specifies the length, or number of bytes, that you want to convert. The conversion starts at the position that STARTPOS defines and continues for the length that the LENGTH parameter specifies. The default value is the record length of the current record. A valid value is an actual numeric value or the name of a field containing a numeric value. The value that you specify for LENGTH plus the numeric value of STARTPOS must be less than or equal to the length of the current record.

Operation — Inline

STARTPOS and LENGTH are optional parameters. If you want to convert the entire active input file, do not specify STARTPOS and LENGTH. STARTPOS and LENGTH default to values that convert the active input file starting at the first byte and continuing for the length of the input record.

This is extremely useful for variable length files. Since each record may have a different length, allowing these parameters to default assures the conversion of all bytes of all records. Bytes that cannot be converted are changed to hexadecimal '07'.

CONVAE can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

The DBFILE parameter identifies CONVAE as a routine that can access database files. This parameter is optional, and you do not have to specify it for nondatabase use. However, you must specify this parameter when using CONVAE in a database application.

Note: The specification of the STARTPOS parameter will differ when you use CONVAE in a database application.

Example

Following is an example of CONVAE.

Input

```
FILE ASCII
  Field-name ...
  ...
FILE EBCDIC ...
  Field-name ...
  ...
JOB INPUT ASCII
%CONVAE
PUT EBCDIC FROM ASCII
```

Results

This example demonstrates the conversion of a file from ASCII to EBCDIC and the subsequent creation of the converted file. The JOB INPUT ASCII statement defines the active input file. Each record is read from the ASCII file and all bytes are converted by the CONVAE routine. The PUT statement writes the converted data to the EBCDIC file.

CONVEA

The CONVEA routine converts EBCDIC alphanumeric characters to their ASCII equivalent. CONVEA is an inline routine that you can use with other CA-PanAudit Plus Routines O-R CA-Easytrieve Plus logic. The inline design of CONVEA reduces execution time and reduces requirements for temporary or permanent storage space. You can also use CONVEA to create a permanent file of converted data.

For conversion from ASCII to EBCDIC, see the CONVAE routine.

Syntax

```
%CONVEA [DBFILE] [STARTPOS identifier] [LENGTH value]
```

[DBFILE]

Specify this optional parameter only for database use of CONVEA. Specify only the literal DBFILE, not the name of the active input file.

[STARTPOS identifier]

This optional parameter specifies the starting position for the conversion process. The identifier must be a previously defined field.

Conversion takes place in the active input file starting at the position defined by STARTPOS and continuing for the length specified by the length parameter.

The default value for STARTPOS is the first byte of the active input file. This means that when using the default value the conversion begins with the first byte of the active input file.

The requirements for specifying STARTPOS are different when you use CONVEA in a database application. In this case the STARTPOS parameter is no longer optional—it is a required parameter.

`[LENGTH value]`

This optional parameter specifies the length, or number of bytes, that you want to convert. The conversion starts at the position that STARTPOS defines and continues for the length that the LENGTH parameter specifies. The default value is the record length of the current record. A valid value is an actual numeric value or the name of a field containing a numeric value. The value that you specify for LENGTH plus the numeric value of STARTPOS must be less than or equal to the length of the current record.

Operation — Inline

STARTPOS and LENGTH are optional parameters. If you want to convert the entire active input file, do not specify STARTPOS and LENGTH. STARTPOS and LENGTH default to values that convert the input file starting at the first byte and continuing for the entire length of the input record.

This is extremely useful for variable length files. Since each record may have a different length, allowing these parameters to default assures the conversion of all bytes of all records. Bytes that cannot be converted are changed to hexadecimal '7F'.

CONVAE can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

The DBFILE parameter identifies CONVEA as a routine that can access database files. This parameter is optional, and you do not have to specify it for nondatabase use. However, you must specify this parameter when using CONVEA in a database application.

Note: The specification of the STARTPOS parameter will differ when you use CONVEA in a database application.

Example

The following is an example of CONVEA.

Input

```
FILE EBCDIC ...
  Field-name ...
  ...
FILE ASCII ...
  Field-name ...
  ...
JOB INPUT EBCDIC
%CONVEA
PUT ASCII FROM EBCDIC
```

Results

This example demonstrates the conversion of a file from EBCDIC to ASCII and the subsequent creation of the converted file. The JOB INPUT EBCDIC statement defines the active input file. Each record is read from the EBCDIC file and is converted by the CONVEA routine. The PUT statement writes the converted data to the ASCII file.

Chapter 5 Generalized/Statistical Routines D-E

This chapter lists alphabetically, and gives detailed descriptions of, routines DATECALC through EXPO.

DATECALC

The DATECALC routine adds or subtracts a given number of days from the date specified in a field and writes the resulting date to a second field.

Syntax

```
%DATECALC date1 format1 {PLUS } days date2 format2 [THRESHOLD value]
                        {MINUS}
```

`date1`

Specify the name of the field containing the date to which a given number of days are to be added or subtracted. The date in this field must be in the format specified by format1. A valid name is any previously defined field.

`format1`

Specify the format of the date1 field. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of date1 is not checked for a valid date with format1. However, CC always maintains the value specified in accordance with the THRESHOLD parameter. If you want date validation, use the DATEVAL routine before using DATECALC. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

```
{PLUS }
{MINUS}
```

Specify whether the value of the days parameter is added to (PLUS) or subtracted from (MINUS) date1.

Note: DATECALC performs an arithmetic calculation. If you specify the PLUS keyword, and the value of the days is negative, the value is subtracted. Conversely, if you specify MINUS, and days is negative, the value is added.

`days`

Specify a numeric literal or a field that contains the value to be added or subtracted.

`date2`

Specify the name of the field to which the resulting date is written. The date is written using the format specified by the format2 parameter. A valid name is any previously defined field.

`format2`

Specify the format for date2.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DATECALC is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

DATECALC generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use DATECALC with database files.

Example

The following is an example of DATECALC.

Input

```
FILE ...
  INVOICE-DATE   1  6  N
  DUE-DATE       W  5  N
  ...
JOB ...
%DATECALC INVOICE-DATE MMDDYY PLUS 30 DUE-DATE YYDDD
...
```

Results

The date contained in the field INVOICE-DATE is increased by 30 days, and the resulting date is converted from Gregorian format to Julian format and is written to the field DUE-DATE.

DATECONV

The DATECONV routine converts a date in one format to any other date format. For example, you can convert month-day-year to year-month-day, Julian to Gregorian, and similar date conversions.

Note: Using non-numeric data or a zero for date fields results in a PAP299 error message which displays the field in error, along with its contents. Execution of the program stops, and a return code 32 is generated.

Syntax

```
%DATECONV date1 format1 date2 format2 [THRESHOLD value]
```

`date1`

Specify the name of the field containing the date to be converted. The date in this field must be in the format specified by format1. The name of any previously defined numeric field is valid.

`format1`

Specify the format of the date1 field. Format1 is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of date1 is not checked for a valid date with the specified format. However, CC always maintains the value specified in accordance with the THRESHOLD parameter. If you want date validation, use the DATEVAL routine before using DATECONV.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

Note: For non-Julian dates, format1 must include the values MM, DD, and YY (in any order). The only valid Julian format is YYDDD.

`date2`

Specify the name of the field to which the converted date will be written. The date is written in the format specified by format2. A valid name is any previously defined field.

`format2`

Specify the format for the date2 field.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DATECONV is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

DATECONV generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use DATECONV with database files.

Example

The following is an example of DATECONV.

Input

```
FILE ...
       JULIAN-DATE     1   5   N
       GREG-DATE       W   6   N
       ...
     JOB ...
     %DATECONV JULIAN-DATE YYDDD GREG-DATE YYMMDD
     ...
```

Results

The Julian date in the field named JULIAN-DATE is converted to Gregorian format, and the result is stored in the field named GREG-DATE.

DATEGEN

The DATEGEN routine generates a date in a given format and writes it to the field parameter.

Note: The FILEGEN routine must be invoked to use the DATEGEN routine.

Syntax

```
%DATEGEN field format [threshold] {BETWEEN  date1 date2          }
                                  {SEQUENCE date1 date2 increment}
```

`field`

Specify the name of the field where DATEGEN places the date it generates. The format of the generated date is specified by the format parameter. Valid names include any field defined in the file name specified in FILEGEN.

`format`

Specify the format of the date field. Format is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = date
YY = year
CC = century
```

Values for MM are limited to the numbers 01 through 12, values for DD are limited to 01 through 31, values for YY are limited to 00 through 99.

If values outside these limits are specified, the numbers will range only between the limits listed previously. YY must be specified whenever CC is specified. CC always maintains the value specified in accordance with the THRESHOLD parameter (on the following page). You can specify MM, DD, YY, and CC in any order.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`[threshold]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DATEGEN is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

`{BETWEEN date1 date2}`

The BETWEEN keyword causes a random date to be generated in the date span you specify by the date1 and date2 parameters. Date1 and date2 must be in the format specified by the format parameter. Dates generated are written to the field specified in the field parameter. Valid values for date1 and date2 are either actual numeric values or the name of a field containing the value. The seed for the random generation of dates is obtained from FILEGEN.

Values generated are greater than or equal to the minimum and less than the maximum. If the desired range of dates is 010184 (MMDDYY) through 061684, specify a minimum of 010184 and a maximum of 061684.

`{SEQUENCE date1 date2 increment}`

The SEQUENCE keyword causes a fixed set of dates to be generated. The set of dates begins with date1 and is incremented for each record by the number of days specified by increment, until date2 is equaled or exceeded. The sequence is then repeated beginning with date1 until a date has been generated for each record being created.

Date1 and date2 must be in the format specified by the format parameter. Valid values for date1, date2, and increment are actual numeric values or the name of a field containing the value. To generate a decreasing set of dates, specify a date1 value greater than the date2 value and code a negative increment.

Operation — Inline

Use DATEGEN only after you have specified the FILEGEN routine. DATEGEN can be specified with ALPHAGEN, NUMGEN, and BADGEN. Conditional execution of DATEGEN is discussed in Conditional Execution of Data Generation Routines following the BADGEN routine.

Operation — Database

DATEGEN, with FILEGEN, cannot be used in a database application.

Examples

The following example illustrates how DATEGEN is used with FILEGEN:

```
FILE CENTFILE F(24)
  NAME    1 12 A
  EMP#   13  5 N
  BIRTH  18  6 N
*
JOB INPUT NULL
  %FILEGEN CENTFILE 25 5 NOHEX
  %ALPHAGEN NAME ‘ CUST ‘ CONSTANT ‘JOHNSON,SMITH,PETERS’
```

```
%NUMGEN EMP# SEQUENCE 1 10050 1
%DATEGEN BIRTH MMDDYY 0  BETWEEN 010151 010175
```

The following examples demonstrate two uses of DATEGEN. An example of the full facilities of the test data generation routines is found following the BADGEN routine.

Between

This example generates a random date between 010120 and 122960 for each record being created and writes it to the BIRTH field.

```
%DATEGEN BIRTH MMDDYY 0 BETWEEN 010120 123060
```

Sequence

A date of 12011999 is written to the EMPLOYED field of the first record, 12211999 to the second record, 01102000 to the third record, 01302000 to the fourth record, etc. Each date is incremented by 20 days until the date 12012001 is equaled or exceeded. The sequence is then repeated beginning with the date 12011999.

```
%DATEGEN EMPLOYED MMDDCCYY 50 SEQUENCE 12011999 12012001 20
```

DATEVAL

The DATEVAL routine examines the content of a specified date field for a valid date in accordance with a specified date format. If the date field contains a valid date, the field DATEVAL-FLAG is set to the value YES. If the date field is invalid, the DATEVAL-FLAG is set to the value NO.

Syntax

```
%DATEVAL field format [THRESHOLD value]
```

`field`

Specify the name of the field that contains the date being validated. Valid names include any previously defined numeric field.

`format`

The format for the comparison is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

You can specify the letter pairs in any order. YY must be specified whenever you specify CC. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DATEVAL is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

The date field is compared to the specified format. For the comparison to be valid, the respective MM, DD, YY, and CC fields must contain valid values. For example, if the date field contains 043184, and the format field contains MMDDYY, the comparison is invalid because the DD (day) portion of the date exceeds 30 for the month of April. If the date field contains 022979, the comparison is invalid because 1979 is not a leap year.

DATEVAL does not produce a report. If the date field contains a valid date, an internal field DATEVAL-FLAG is set to the value YES. If the date field is invalid, the DATEVAL-FLAG is set to the value NO.

To perform further processing activities, you must code CA-Easytrieve Plus logic following the invocation of DATEVAL. You can code IF statements that test the DATEVAL-FLAG field.

For example, you may want to print a report of invalid dates, write all records with valid dates to an output file and perform further processing of the invalid dates, or any combination of events. See the CA-Easytrieve Plus *Reference Guide* for coding techniques. The example demonstrates coding using IF, DISPLAY, and END-IF statements.

Operation — Database

No change in the specification of parameters is required to use DATEVAL with database files.

Example

The following is an example of DATEVAL.

Input

```
FILE ...
  DATE          1  6  N
  INVOICE-NUM   7  4  P
  ...
JOB ...
%DATEVAL DATE MMDDYY
IF DATEVAL-FLAG EQ 'NO'
  DISPLAY +5 INVOICE-NUM +5 DATE
END-IF
...
```

Output

This example prints the invoice number and date for every record with an invalid date according to the format MMDDYY.

```
2983     083781
3953     023072
4263     063184
5337     131278
7654     000000
```

DAYSAGO

The DAYSAGO routine calculates the number of days that have elapsed between the current date and the date in a specified field. The result is compared to a value with a relational operator. If the calculated number of days satisfies the specified condition, the internal flag DAYSAGO-FLAG is set to the value YES. If the condition is not satisfied, DAYSAGO-FLAG is set to the value NO.

Note: DAYSAGOL functionally replaces DAYSAGO and supports a wider range of dates accurately. You should use DAYSAGOL in place of DAYSAGO in systems running CA-Easytrieve+ 6.2 or above.

Syntax

```
%DAYSAGO datefield format operator value [THRESHOLD value]
```

`datefield`

Specify the name of the field containing the date used for calculating the number of elapsed days. The date in this field must be in the format specified by the format parameter. A valid name is any previously defined field.

`format`

Specify the format of datefield. Format is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of datefield is not checked for a valid date with the specified format. If you want date validation, use the DATEVAL routine before using DAYSAGO. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`operator`

Specify any relational operator (EQ, NE, LT, LE, GT, or GE).

`value`

The number of days between the date in datefield and the current date is compared to this numerical value. For value, you can specify an actual numeric value or the name of a field containing the numeric value.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DAYSAGO is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

DAYSAGO does not produce a report. If the calculated number of days satisfies the condition specified, the internal flag DAYSAGO-FLAG is set to the value YES. If the condition is not satisfied, DAYSAGO-FLAG is set to the value NO.

To perform further processing activities, you must code CA-Easytrieve Plus logic following the invocation of DAYSAGO. Code IF and END-IF statements, which test the DAYSAGO-FLAG field around the logic you want to perform. The example demonstrates this coding technique.

Operation — Database

No change in the specification of parameters is required to use DAYSAGO with database files.

Example

The following is an example of DAYSAGO.

Input

```
FILE ...
  EMPLOYEE-NAME     1  20   A
  SSN              21  11   A
  HIRE-DATE         32   6   N
  ...
JOB ...
%DAYSAGO HIRE-DATE MMDDYY LE 120
IF DAYSAGO-FLAG EQ 'YES'
     PRINT DAYSAGO-REPORT
END-IF
...
REPORT DAYSAGO-REPORT
TITLE 1 'EMPLOYEES HIRED SINCE JANUARY 1984'
LINE EMPLOYEE-NAME SSN HIRE-DATE
```

Output

```
     EMPLOYEES HIRED SINCE JANUARY 1984

    EMPLOYEE-NAME          SSN      HIRE-DATE

   PETER    BRENNON    324-23-5467  022484
 CHARLES     JENSEN    342-23-0232  031284
   BETTY     WALTON    384-64-8547  010384
JENNIFER     WILSON    311-42-7472  033084
```

This example examines a file to list all employees hired in the last 120 days. The DAYSAGO-FLAG is examined, and a report listing the employee name, social security number, and date hired is printed if DAYSAGO-FLAG contains the value YES.

DAYSAGOL

The DAYSAGOL routine calculates the number of days that have elapsed between the current date and the date in a specified field. The result is compared to a value with a relational operator. If the calculated number of days satisfies the specified condition, the internal flag DAYSAGO-FLAG is set to the value YES. If the condition is not satisfied, DAYSAGO-FLAG is set to the value NO.

Note: DAYSAGOL functionality replaces DAYSAGO and supports a wider range of dates accurately. We recommend that you use DAYSAGOL in place of DAYSAGO in systems running CA-Easytrieve Plus 6.2 or above.

Syntax

```
%DAYSAGOL datefield format operator value [THRESHOLD value]
```

`datefield`

Specify the name of the field containing the date used for calculating the number of elapsed days. The date in this field must be in the format specified by the format parameter. A valid name is any previously defined field.

`format`

Specify the format of datefield. Format is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of datefield is not checked for a valid date with the specified format. If you want date validation, use the DATEVAL routine before using DAYSAGOL. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`operator`

Specify any relational operator (EQ, NE, LT, LE, GT, or GE).

`value`

The number of days between the date in datefield and the current date is compared to this numerical value. For value, you can specify an actual numeric value or the name of a field containing the numeric value.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DAYSAGOL is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

DAYSAGOL does not produce a report. If the calculated number of days satisfies the condition specified, the internal flag DAYSAGO-FLAG is set to the value YES. If the condition is not satisfied, DAYSAGO-FLAG is set to the value NO.

To perform further processing activities, you must code CA-Easytrieve Plus logic following the invocation of DAYSAGOL. Code IF and END-IF statements, which test the DAYSAGO-FLAG field around the logic you want to perform. The example demonstrates this coding technique.

Operation — Database

No change in the specification of parameters is required to use DAYSAGOL with database files.

Example

The following is an example of DAYSAGOL.

Input

```
FILE ...
  EMPLOYEE-NAME     1  20   A
  SSN              21  11   A
  HIRE-DATE         32   6   N
  ...
JOB ...
%DAYSAGOL HIRE-DATE MMDDYY LE 120
IF DAYSAGO-FLAG EQ 'YES'
     PRINT DAYSAGOL-REPORT
END-IF
...
REPORT DAYSAGOL-REPORT
TITLE 1 'EMPLOYEES HIRED SINCE JANUARY 1984'
LINE EMPLOYEE-NAME SSN HIRE-DATE
```

Output

```
     EMPLOYEES HIRED SINCE JANUARY 1984

    EMPLOYEE-NAME          SSN      HIRE-DATE

   PETER    BRENNON    324-23-5467  022484
 CHARLES     JENSEN    342-23-0232  031284
   BETTY     WALTON    384-64-8547  010384
JENNIFER     WILSON    311-42-7472  033084
```

This example examines a file to list all employees hired in the last 120 days. The DAYSAGO-FLAG is examined, and a report listing the employee name, social security number, and date hired is printed if DAYSAGO-FLAG contains the value YES.

DAYSCALC

The DAYSCALC routine calculates the number of elapsed days between two specified dates of any format. The calculation is:

```
days = date1 - date2
```

Syntax

```
%DAYSCALC date1 format1 date2 format2 days [THRESHOLD value]
```

`date1`

Specify the name of the field containing the date that date2 is subtracted from. The date in this field must be in the format specified by format1. A valid name is any previously defined field.

`format1`

Specify the format of the date1 field. Format1 is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of date1 is not checked for a valid date with the specified format. If you want date validation, use the DATEVAL routine before using DAYSCALC. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`date2`

Specify the name of the field containing the date to be subtracted. The date in this field must be in the format specified by format2. A valid field name is any previously defined field.

`format2`

Specify the format for the date2 field.

`days`

Specify the name of the numeric field that will hold the results of the calculation. The value of days is negative if date1 is earlier than date2. However, a negative indicator for days is printed only if it is defined with decimal positions (0 through 18) or with a user-defined edit mask containing a negative number indicator. For additional information on edit masks, see the CA-Easytrieve Plus *Reference Guide.*

You must define the days field before invoking DAYSCALC.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD default value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DAYSCALC is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

DAYSCALC generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Use DAYSCALC only on dates during the years 1800 through 2199.

Operation — Database

No change in the specification of parameters is required to use DAYSCALC with database files.

Example

The following is an example of DAYSCALC.

Input

```
FILE ...
  END-DATE        1   6   N
  START-DATE      7   6   N
  ...
RESULT            W   10  P   0
JOB ...
%DAYSCALC END-DATE MMDDYY START-DATE MMDDYY RESULT
...
```

Results

The number of elapsed days between the fields END-DATE and START-DATE is calculated, and the result is stored in a field called RESULT. RESULT is defined with 0 decimal places to ensure that a negative sign will print if the field is negative.

DECRYPT

The DECRYPT routine decrypts data from a previously encrypted file. The values that you specify for key, STARTPOS, and LENGTH must be identical to the values specified when the file is encrypted.

DECRYPT is an inline routine that you can use with other CA-PanAudit Plus Routines O-R CA-Easytrieve Plus logic. The inline design of DECRYPT reduces execution time and reduces the requirements for temporary or permanent storage space. You can also use DECRYPT to create a permanent or temporary file of decrypted data.

Note: The ENCRYPT and DECRYPT routines are not compatible across releases of CA-PanAudit Plus or CA-PanAudit.

Syntax

```
%DECRYPT key [STARTPOS identifier] [LENGTH value]
```

`key`

Specify the key value that is used when the file is encrypted. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits long with no decimal places. Values greater than seven digits are truncated on the left.

`[STARTPOS identifier]`

This optional parameter specifies the starting position for the decryption process. The identifier can be the name of a field or a numeric value.

If identifier is a field name, the decryption process starts at the first byte of the field located in the active input file. In this case, a valid value for identifier is any field defined in the active input file.

If identifier is a numeric value, the decryption process starts at the byte location that identifier specifies. For example, if identifier is the numeric value five, decryption begins with the fifth byte of the active input file. In this case, a valid value for identifier is an actual numeric value only.

The default value for STARTPOS is the numeric value one, which means that the decryption begins with the first byte of the active input file.

Decryption takes place starting at the file position that STARTPOS defines and continues for the length that the length parameter specifies. The value for STARTPOS must be the same value specified when the file was encrypted.

The requirements for the specification of STARTPOS are different when you use DECRYPT in a database application. In this case the STARTPOS parameter is no longer optional; it is a required parameter. Furthermore, under these circumstances, STARTPOS cannot be a numeric value; it can only be a field name in a database record defined in the library section.

`[LENGTH value]`

This optional parameter specifies the length, or number of bytes to be decrypted. The decryption process starts at the position that STARTPOS defines and continues for the length that the LENGTH parameter specifies. The default value for LENGTH is the record length of the current record. The value for LENGTH must be the same value specified when the file was encrypted. A valid value is an actual numeric value or the name of a field containing a numeric value. The value that you specify for LENGTH plus the numeric value of STARTPOS must be less than or equal to the length of the current record.

Operation — Inline

STARTPOS and LENGTH are optional parameters. If the entire input file is encrypted by allowing STARTPOS and LENGTH to assume their default values, they can also be allowed to default during the decryption process. The default values decrypt the entire active input file.

This is extremely useful for variable length files. Because each record can have a different length, allowing these parameters to default ensures the decryption of all bytes of all records.

Because the encryption process renders the data unintelligible, it is important to remember the key, STARTPOS, and LENGTH values that ENCRYPT specifies. The values that you specify for these parameters in the DECRYPT routine must be identical to the values specified during encryption.

DECRYPT performs the decryption starting at the byte that the STARTPOS parameter defines. Each byte is decrypted for the specified length according to the value that the key parameter specifies. The decrypted information is moved to the input buffer of the active input file. This allows the decryption process to occur without creating a temporary copy of the data. You can also decrypt data and use it with other CA-PanAudit Plus Routines O-R CA-Easytrieve Plus logic, all in the same job activity.

DECRYPT does not allow screening of input data due to the possibility of invalid decryption of encrypted data. An encrypted file must be decrypted in its entirety, even if you only want a portion of the records.

Operation — Database

Notice the difference in the specification of the STARTPOS parameter when DECRYPT is used in a database application.

Examples

The following are two examples of the DECRYPT routine.

Example One

Input

```
FILE INFILE  FB (200 10000)
  UNIT-PRICE   10   4   P  2
  QUANTITY     14   4   P  0
  ...
  DEFINE TOTAL ...
JOB INPUT INFILE
%DECRYPT 139743 LENGTH 100
     .
   TOTAL = UNIT-PRICE * QUANTITY
     .
```

Results

This example is an application that demonstrates the inline ability of DECRYPT. Each record is read, and the DECRYPT routine decrypts bytes 1 - 100 with the key value of 139743. Since the location of the fields UNITPRICE and QUANTITY is in the first 100 bytes, their values are decrypted and written to the input buffer. Therefore, when these fields are subsequently multiplied to produce a value for TOTAL, the decrypted values are used.

This process continues until all records are read, decrypted, and processed. After the job is finished, all decrypted information is lost because no temporary or permanent files were created. In this way, the DECRYPT routine ensures the continued integrity of all encrypted information.

Example Two

Input

```
FILE INFILE  FB (200 20000)
  TARGET-BYTE      100   1   A
  ...
FILE OUTFILE  FB (200 20000)
  Field-name ...
  ...
%DECRYPT 7623 STARTPOS TARGET-BYTE LENGTH 101
PUT OUTFILE FROM INFILE
```

Results

This example demonstrates the decryption of the input file and the subsequent creation of the original file. Each record is read, and the DECRYPT routine decrypts bytes 100 through 200 with the key value 7623. After the record is decrypted, it is written to OUTFILE by the PUT statement.

Do not use this method for normal processing of encrypted data because the decrypted information exists as a permanent file that increases the possibility of accidental or intentional disclosure. This defeats the purpose of cryptography.

DISCPCT

The DISCPCT routine calculates the percent of a file's total records that constitutes a representative discovery sample. The calculation for sample size is based on three statistical parameters:

- File size
- Expected error rate
- Expected probability that the sample contains at least one error

A report lists the input parameters and the calculated sample percentage.

Use the DISCSMP routine to calculate the appropriate sample size and then randomly select the records from a file.

Syntax

```
%DISCPCT size error probability
```

`size`

Specify the total number of records in the population being examined. A valid value is an actual numeric value or the name of a field containing a numeric value.

`error`

Specify the percentage of error that you estimate will be found as a result of the test. Your judgment of the probable error rate can be guided by the results of a previous test, a preliminary survey, or a small pilot test of transactions. A valid value for error is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`probability`

Specify a number, expressed as a percentage, that designates the probability that at least one error will be found in the resulting sample. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

Operation — Inline

DISCPCT provides the ability to study the results of discovery sampling without reading an input file. It provides a technique for studying results so that you can make the best possible parameter selection for the given application. It can be invoked any number of times to demonstrate the effects of varying the input parameters (see Example). For example, you can use it to evaluate the effects on the sample size of varying the probability percentage.

If DISCPCT is used with a JOB INPUT NULL statement, the job must contain a STOP statement. When satisfactory results are obtained, the DISCSMP routine can be used to produce the identical results and to randomly select the desired samples from a file.

Operation — Database

No change in the specification of parameters is required to use DISCPCT with database files.

Example

The following is an example of DISCPCT.

Input

```
JOB INPUT NULL
...
%DISCPCT 100000 5.0 70.00
%DISCPCT 100000 5.0 90.00
%DISCPCT 100000 5.0 99.99
...
STOP
```

Output

This example demonstrates a technique for evaluating the effects on the sample percentage by varying the probability. The input consists of three invocations of DISCPCT with identical values for population size and error rate, but with probability percentages of 70, 90, and 99.99. As the demand increases for the probability that an error exists in the sample, the required sample percentages increase. The report lists the input values and the different sample percents for the three values specified.

```
                DISCOVERY SAMPLING REPORT

POPULATION SIZE     ERROR RATE   PROBABILITY    SAMPLE PERCENT
        100,000        5.0          70.00             .0240

POPULATION SIZE     ERROR RATE   PROBABILITY    SAMPLE PERCENT
        100,000        5.0          90.00             .0460

POPULATION SIZE     ERROR RATE   PROBABILITY    SAMPLE PERCENT
        100,000        5.0          99.99             .1840
```

DISCSMP

The DISCSMP routine calculates the percent of a file's total records that constitutes a representative discovery sample and then randomly selects the appropriate number of records from the file. The calculation for sample size is based on three statistical parameters:

- File size
- Expected error rate
- Expected probability that the sample contains at least one error

Records selected can be written to an output file. A report lists the input parameters and the result of the sample size calculation.

Use the DISCPCT routine to calculate the appropriate sample size without selecting records.

Syntax

```
%DISCSMP1 infile size error probability seed

%DISCSMP2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to DISCSMP. A valid name is any previously defined file.

`size`

Specify the total number of records in the population being examined. A valid value is an actual numeric value or the name of a field containing a numeric value.

`error`

Specify the percentage of error you estimate will be found as a result of the test. Your judgment of the probable error rate can be guided by the results of a previous test, a preliminary survey, or a small pilot test of transactions. A valid value for error is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`probability`

Specify a number, expressed as a percentage, that designates the probability that at least one error will be found in the resulting sample. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`seed`

Specify an arbitrary number that initiates the random number generator. This seed is used to randomize the selection of samples from the file. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are to be written to an output file.

OUTFILE—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

`[DBFILE infile]`

This optional parameter specifies a database file for use with DISCSMP. Infile identifies the name of the input file to DISCSMP. The name must be the same name that you specified for infile on the first invocation statement.

`[PERFORM procname]`

Specify the name of a CA-Easytrieve Plus procedure that is performed by the DISCSMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field DISCSMP-SELECTED is set to the value YES. If a record is not selected, DISCSMP-SELECTED is set to the value NO.

After the invocation of DISCSMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the DISCSMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that DISCSMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "**Advanced Techniques**."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You can adjust the size parameter when screening code is inserted that causes records to be bypassed from DISCSMP processing. The value specified for the size parameter must represent the size of the population being examined for the discovery sampling. For the resulting sample percentage and optional sample file to be accurate, the size parameter must be adjusted by the number of records that are bypassed.

For example, if you specify 50,000 as the file size, and screening code causes 25,000 of these records to be bypassed, the population size sampled by DISCSMP will actually be 25,000. The calculated percentage therefore is incorrect, and the sample file created is also invalid.

To avoid this result, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent a sample file from being created. Notice the number of records processed by DISCSMP in the report. You can rerun the job using the record count listed in the report for the size parameter and specify an output file name in place of NOFILE. This ensures the correct results from DISCSMP processing, while bypassing unwanted records.

Operation — Database

The DBFILE parameter identifies DISCSMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using DISCSMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of DISCSMP.

Example

The following is an example of DISCSMP.

Input

```
FILE INFILE FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE (44 4400)
%DISCSMP1 INFILE 10000 10.0 99.0 93843
%DISCSMP2 OUTFILE
```

Output

The report lists the input parameters followed by the results of the DISCSMP calculations. The report also provides the results of the random sampling process, including whether an output file is created.

```
                    DISCOVERY SAMPLING REPORT

                        INPUT PARAMETERS

          INPUT FILENAME                         INFILE
          TOTAL POPULATION SIZE                  10,000
          EXPECTED ERROR RATE                        10.00%
          PROBABILITY OF SINGLE ERROR OCCUR          99.00%
```

```
                    SAMPLE RESULTS

SAMPLE PERCENTAGE REQUIRED                   0.4594%
SAMPLE SIZE REQUIRED                        46

                    SAMPLE FILE

NUMBER OF RECORDS PROCESSED             10,000
NUMBER OF RECORDS REQUESTED                 46
NUMBER OF RECORDS IN SAMPLE FILE            46

FILE OUTFILE WILL BE CREATED
```

DIVIDE

The DIVIDE routine calculates the integer quotient and remainder of two numbers. For example, 6/4 = 1 remainder 2. This routine is not intended to replace the CA-Easytrieve Plus division operation (/), but is intended primarily to produce a remainder in those cases in which it is of significance.

Syntax

```
%DIVIDE    number    divisor    quotient    remainder
```

`number`

Specify the number to be divided. It can be actual numeric value or a previously defined field containing a numeric value.

`divisor`

Specify the value by which the number is divided. This must be a nonzero integer. It can be a numeric value or a previously defined field containing a numeric value.

`quotient`

Specify the field to which the quotient of the calculation is written. A valid name is any previously defined numeric field.

`remainder`

Specify the field to which the remainder of the calculation is written. A valid name is any previously defined numeric field.

Operation — Inline

DIVIDE is designed for use with integer fields only. Results of the calculation are incorrect if decimal places are defined.

DIVIDE generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in specification of parameters is required to use DIVIDE with database files.

Example

The following is an example of DIVIDE. In this example, quotients and remainders were calculated successfully until the field DI contained a zero.

Input

```
FILE INFILE
  NU        1 2 N
  DI        4 2 N
*
DEFINE QU    W 4 N
DEFINE RE    W 4 N
*
JOB INPUT INFILE
*
%DIVIDE NU DI QU RE
  DISPLAY  NU ' / ' DI ' = '  QU ' REMAINDER ' RE
```

Output

```
  02 / 04 = 0000 REMAINDER 0002
  12 / 03 = 0004 REMAINDER 0000
  07 / 02 = 0003 REMAINDER 0001
  09 / 01 = 0009 REMAINDER 0000
  09 / 02 = 0004 REMAINDER 0001
  09 / 03 = 0003 REMAINDER 0000
  09 / 04 = 0002 REMAINDER 0001
  09 / 05 = 0001 REMAINDER 0004
  09 / 06 = 0001 REMAINDER 0003
  09 / 07 = 0001 REMAINDER 0002
  09 / 08 = 0001 REMAINDER 0001
  09 / 09 = 0001 REMAINDER 0000
  09 / 89 = 0000 REMAINDER 0009
  09 / 92 = 0000 REMAINDER 0009
  73 / 09 = 0008 REMAINDER 0001
  00 / 12 = 0000 REMAINDER 0000
  07 / 40 = 0000 REMAINDER 0007
  01 / 04 = 0000 REMAINDER 0001
  17 / 13 = 0001 REMAINDER 0004
  06 / 04 = 0001 REMAINDER 0002
  *****  PAP308  DIVISION BY ZERO   DI
```

DOLUNIT

The DOLUNIT routine performs a dollar unit sampling of the input file. DOLUNIT selects records for sampling according to monetary units rather than physical attributes. It optionally creates a sample file based on the selections made during dollar unit processing. You can print a report including the input parameters, the makeup of the sample file, and the positive, negative, and total book values of the file.

Syntax

```
%DOLUNIT1 infile field width cutoff seed [VALUE {ABS}]   [REPORT {YES}]
                                         [      {ACT}]   [       {NO }]
                                         [      {POS}]   [       {   }]

%DOLUNIT2 {outfile} [TOP {topfile}] [KEY {keyfile}] [DBFILE infile] +
          {NOFILE } [    {NOFILE }] [    {NOFILE }]

                    [PERFORM procname]
```

`infile`

Specify the name of the input file to DOLUNIT. A valid name is any previously defined file.

`field`

Specify the name of the quantitative field from which the values for the dollar unit sampling are taken. A valid name is any quantitative field defined in the input file.

`width`

Specify the value of the cell width for the sample. This determines the target value that must be exceeded for a record to be selected for the sample file. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places. Values greater than two decimal places are truncated on the right. (Cell width is analogous to the target value parameter of the SPS routine.)

`cutoff`

Specify the value for cutoff for the top stratum. Any record having a value greater than or equal to the cutoff becomes part of the top stratum and is sent to the sample file. To direct the top stratum samples to a separate file, use the TOP parameter. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places. Values with greater than two decimal places are truncated on the right.

```
seed
```

Specify an arbitrary number that initiates the random number generator. This seed is used to randomize the updated target value for each cell. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
[VALUE{ABS}]
[     {ACT}]
[     {POS}]
```

This optional parameter controls the value for the input field used in the sampling process. The default value is ABS.

ABS—Specifies that the absolute value of the input field is used in the sampling process. This method places equal emphasis on both positive and negative values.

ACT—Specifies that the actual value of the input field is used in the sampling process. This method places emphasis on the net value of positive and negative values.

POS—Specifies that only the values of the input field that are greater than zero are used in the sampling process. This method places emphasis on the positive values.

```
[REPORT{YES}]
[      {NO }]
```

This optional parameter specifies whether the DOLUNIT report is produced. Specify NO to inhibit the printing of the report. The default is YES, which produces the report.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are to be written to an output file.

outfile—Records selected for the sample are written to the file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
[TOP {topfile}]
[    {NOFILE }]
```

topfile—This optional parameter specifies the name of the file to which the top stratum records (audited values that exceed the cutoff) are written. File characteristics must be coded on the FILE statement for this output file. Topfile must have the same file characteristics as the input file, or topfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for topfile include any previously defined file. The default is for top stratum records to be written to the file indicated by the outfile parameter. For use of this parameter, see Operation — Stand-alone DISPLAY in this routine.

NOFILE—This option is valid only for database use of DOLUNIT. NOFILE specifies that top stratum records are to be written to the file indicated by the outfile parameter.

```
[KEY {keyfile}]
[    {NOFILE }]
```

keyfile—Key records are those records known to be significant (such as errorprone records or records with an unusual history or specific values). This optional parameter specifies the name of the file to which key records are to be written. Use of this parameter prevents key records from appearing on the sample file. File characteristics must be coded on the FILE statement for this output file. Keyfile must have the same file characteristics as the input file, or keyfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for keyfile include any previously defined file. For use of this parameter, see Operation — Stand-alone DISPLAY in this routine.

NOFILE—This option is valid only for database use of DOLUNIT. NOFILE specifies that no key processing will occur.

```
[DBFILE infile]
```

Specify this optional parameter only for database use of DOLUNIT. Specify the name of the input file to DOLUNIT. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure that is performed by the DOLUNIT routine after each record is selected or not selected for an output file. If a record is selected for an output file, the internal field DOLUNIT-SELECTED is set to the value YES. If a record is not selected for an output file, DOLUNIT-SELECTED is set to the value NO. The internal field name is DOLUNIT-SELECTED for the output file that the outfile/NOFILE parameter identifies, DOLUTOP-SELECTED for the output file that topfile identifies, and DOLUKEYSELECTED for the output file that keyfile identifies.

After the invocation of DOLUNIT2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for an output file.

For example, the procedure can test the DOLUTOP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all records selected for the topfile in addition to the normal report that DOLUNIT produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "**Advanced Techniques**."

If you specify this parameter, it must follow any occurrence of the TOP or KEY parameters. This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

DOLUNIT can be used without the KEY or TOP options. If a separate file for top stratum items is required, use the TOP parameter with the associated topfile filename. If key items are to be separated into a different file, use the KEY parameter with the associated keyfile parameter.

If both TOP and KEY parameters are specified, the one that is specified first on the macro invocation statement takes precedence. For example, if a record satisfies the conditions for both top stratum and key processing, it is written to the sample file indicated by the parameters TOP or KEY (whichever is specified first).

Key records are identified by using an IF statement. If the KEY parameter is specified, IF, PERFORM DOLUKEY, and END-IF statements must be placed between the %DOLUNIT1 and %DOLUNIT2 statements.

The following example writes all records to the file KEYFILE that have the field EMPLOYEE-CODE equal to 99:

```
FILE INFILE ...
  PAY ...
  EMPLOYEE-CODE ...
  ...
FILE KEYFILE ...
  ...
FILE OUTFILE ...
  ...
%DOLUNIT1 INFILE PAY 10000 2000 1357
IF EMPLOYEE-CODE EQ 99
     PERFORM DOLUKEY
END-IF
%DOLUNIT2 OUTFILE KEY KEYFILE
```

Two sample files are created. Records from the dollar unit algorithm that are sampled are written to the file OUTFILE. Records with EMPLOYEE-CODE equal to 99 are not used in the dollar unit algorithm and are written to KEYFILE.

The following example is the same as the preceding example except that screening logic is added:

```
FILE INFILE ...
  PAY ...
  EMPLOYEE-CODE ...
  ...
FILE KEYFILE ...
  ...
FILE OUTFILE ...
  ...
%DOLUNIT1 INFILE PAY 10000 2000 1357
IF EMPLOYEE-CODE EQ 0
     GO TO JOB
END-IF
IF EMPLOYEE-CODE EQ 99
     PERFORM DOLUKEY
END-IF
%DOLUNIT2 OUTFILE KEY KEYFILE
```

The IF, GO TO JOB, and END-IF statements constitute the screening logic. These statements bypass any records with EMPLOYEE-CODE equal to zero. The IF, PERFORM DOLUKEY, and END-IF statements constitute the key processing logic. It does not matter whether the screening logic is coded before or after the key logic. A record is skipped if it satisfies the conditions for the screening logic. Each set of logic must be coded with the IF statement, followed by the GO TO JOB (screening) or PERFORM DOLUKEY (key processing), and the END-IF statement.

Operation — Database

The DBFILE parameter identifies DOLUNIT as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using DOLUNIT in a database application. Furthermore, you must specify all parameters on the second invocation statement, with the exception of TOP and KEY, in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of DOLUNIT.

When you specify DBFILE, you must specify both TOP and KEY. If you do not want a TOP file, specify NOFILE. This causes all records for the top stratum to be written to the file that the outfile parameter identifies. If you do not want a KEY file, specify NOFILE and do not code a PERFORM DOLUKEY statement. This prevents records from being written to a KEY file.

The requirement that you specify the TOP and KEY parameters applies only to the database use of DOLUNIT.

Example

The following is an example of DOLUNIT.

Input

```
FILE PAYFILE ...
  GROSS ...
  DEPT ...
  ...
FILE OUTFL ...
  ...
FILE TOPFL ...
  ...
FILE KEYFL ...
  ...
%DOLUNIT1 PAYFILE GROSS 1000000 60000 135
IF DEPT EQ 21
     PERFORM DOLUKEY
END-IF
%DOLUNIT2 OUTFL TOP TOPFL KEY KEYFL
```

Output

```
                     DOLLAR UNIT SAMPLING REPORT

                          INPUT PARAMETERS

INPUT FILENAME                                   PAYFILE
INPUT FIELD                                        GROSS
VALUE OF INPUT FIELD IS                              ABS
CELL WIDTH                                  1,000,000.00
TOP STRATUM CUTOFF                             60,000.00

                                SAMPLE FILE(S)

NUMBER OF CELLS PROCESSED                             37


                       RECORD  FILE                         TOTAL
                        COUNT  NAME                         VALUE

NOT SELECTED            1,157                       34,860,801.00

GENERAL SAMPLE             36  OUTFL                 1,319,107.00
TOP STRATUM                 8  TOPFL                   481,743.00
KEY VALUE                  13  KEYFL                   392,251.00
TOTAL SAMPLES              57                        2,193,101.00

PROCESSED TOTAL         1,214                       37,053,902.00


      ABSOLUTE VALUE TOTAL

                  37,046,280.00    POSITIVE BOOK VALUE
 MINUS                 7,622.00-   NEGATIVE BOOK VALUE

                  37,053,902.00    TOTAL    BOOK VALUE - ABSOLUTE


      ACTUAL   VALUE TOTAL

                  37,046,280.00    POSITIVE BOOK VALUE
 PLUS                  7,622.00-   NEGATIVE BOOK VALUE

                  37,038,658.00    TOTAL    BOOKVALUE - ACTUAL
```

The DOLUNIT report:

- Lists the input parameters, the results of the dollar unit sampling, and positive, negative, and total book values of the file.
- Identifies the number of records in all appropriate areas, including the total number of samples.
- Provides the count for key processing, depending on whether the KEY parameter is specified.
- Lists the file names of the general, top, and key sample files.
- Calculates a processed total of all records that participated in the dollar unit sampling. This total is the actual value of those records that participated.

The positive and negative book values are the actual values accumulated by the routine. The negative book values are then subtracted and added to yield the absolute and actual value totals. The value used in the dollar unit sampling algorithm is based upon the VALUE parameter: absolute, actual, or positive.

DUPTEST

The DUPTEST routine tests a field to determine if the content is identical in more than one record. It prints a report of all duplicate records. Two options for output are available:

- A detailed report of each duplicate record, with one field to identify the record and one field to provide additional information.
- A summary report, listing only the total number of duplicate records detected.

With either option, the duplicate records can be written to an output file.

Syntax

```
%DUPTEST1 infile  [LRECL length]

%DUPTEST2 infile {S} field {outfile} {field2 field3}
                 {U}       {NOFILE } {SUMMARY      }
```

`infile`

Specify the name of the input file to DUPTEST. A valid name is any previously defined file.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following example:

```
Infile-lrecl + 1 work byte + 4 RDW bytes = LRECL
```

`{S}`
`{U}`

Specify whether the records input to DUPTEST are sorted or unsorted.

S—Indicates that records are sorted in order by the field parameter. The sorted order can be in ascending or descending sequence.

U—Indicates that records are not in sorted order. DUPTEST will sequence these records in temporary storage in ascending order by the field parameter before it begins the comparison process.

```
field
```

Specify the name of the field for which duplicates are searched. If the content of this field is the same in two or more records, a duplicate condition exists, and the activities specified in subsequent parameters are performed. A valid name is any nonquantitative field defined in the input file.

```
{outfile}
{NOFILE }
```

Specify whether an output file of duplicate records is created.

outfile—Duplicate records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. Valid names for outfile include any previously defined file.

NOFILE—Duplicate records are not written to an output file.

```
{field2 field3}
{SUMMARY      }
```

Specify whether the report will contain a detail line for each duplicate record or a summary report giving the total of duplicates.

field2 field3—When this option is specified, three fields are listed for each duplicate record:

- Field (the field for which duplicates are searched)
- Field2
- Field3

Valid names are any previously defined fields.

SUMMARY—When this option is specified, a summary report is produced. This consists of the total number of sets of duplicates and the total of all duplicate records in the file.

Operation — Stand-alone REPORT

Use the DUPTEST routine to test for duplicate records in a file. A detail or summary report is produced, depending on the option you specify. If specified, duplicate records can be written to an output file.

Operation — Database

DUPTEST can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can either be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of DUPTEST.

Input

```
FILE CUSTFIL ...
  INVNO       1     5     N
  NAME        7    15     A
  BALANCE    23     7     N 2
...
%DUPTEST1 CUSTFIL
%DUPTEST2 CUSTFIL S INVNO NOFILE NAME BALANCE
```

Output

The report lists three sets of records with duplicate invoice numbers. In each case, the total of any numeric field is listed in addition to the tally for each set of duplicates.

```
4/20/88                    REPORT OF DUPLICATE RECORDS     PAGE 1

                                  KEY IS INVNO
                            LISTING NAME AND BALANCE
                               NOFILE IS PRODUCED

           DUPLICATE      DUPLICATE        DUPLICATE     NUMBER OF
           INVNO          NAME             BALANCE      DUPLICATES

            10134         LARRY JONES         123.64
                          MARY JAFFEY         436.34
          INVNO TOTAL                         559.98          2

              63674        CRAIG HALL         563.67
                           CRAIG HILL         563.67
          INVNO TOTAL                       1,127.34          2

              98723       GARY CONDREN       1,504.66
                           KATHY BRADY         564.60
                          REX THOMPSON          44.33
          INVNO TOTAL                         2,113.59          3

          FINAL TOTAL                         3,800.91          7
```

EACHNTH

The EACHNTH routine selects a sample from an existing file, based on a specified starting point and subsequent selection of every nth record. Generally, this method does not produce a statistically valid sample because each record does not have an equal probable selection. However, it is useful in certain types of compliance testing.

Syntax

```
%EACHNTH1    infile  interval  [START num]

%EACHNTH2    {outfile}  [DBFILE infile]  [PERFORM procname]
             {NOFILE }
```

`infile`

Specify the name of the input file to EACHNTH. A valid name is any previously defined file.

`interval`

Specify the interval between records selected from the file. For example, if 5 is specified, every 5th record will be selected. A valid value is an actual numeric value or the name of a field containing the numeric value.

`[START num]`

Use this optional parameter to start selecting records at a record other than 1. A valid value is an actual numeric value or the name of a field containing the numeric value.

```
{outfile}
{NOFILE }
```

Specify whether an output file of selected records is created.

outfile—Selected records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. Valid names for outfile include any previously defined file.

NOFILE—Selected records are not written to an output file.

`[DBFILE infile]`

You must specify this option parameter only for database use of EACHNTH. Specify the name of the input file to EACHNTH. The name must be the same name that was specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure which is performed by the EACHNTH routine after each record is selected or not selected for the sample file. The internal field, EACHNTH-SELECTED, is set to YES if a record is selected for the sample file and to NO if a record is not selected.

After the invocation of EACHNTH2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record was selected for the sample file. For example, the procedure could test the EACHNTH-SELECTED field and display the appropriate fields of the input record if the value is YES. This provides a listing of selected records in addition to the normal report that EACHNTH produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide*. For an example of the use of this parameter, see the chapter "**Advanced Techniques**."

This is an optional parameter. If you do not specify it, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

EACHNTH provides a simple way to create a subset of the records in a file. However, because EACHNTH does not produce a statistically valid sample, it is important that you know the number of records contained in the input file. You set the interval at which records are selected. EACHNTH will select records until the end of the input file is reached.

Operation — Database

The DBFILE parameter identifies EACHNTH as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using EACHNTH in a database application. Furthermore, you must specify all parameters on the second invocation statement, in the order shown in the description of the syntax. This restriction applies only to the database use of EACHNTH.

Example

The following is an example of EACHNTH.

Input

```
FILE PAYFILE FB (44 4400)
*
FILE OUTFILE FB (44 4400)
*
%EACHNTH1 PAYFILE 5 START 4
%EACHNTH2 OUTFILE
```

Output

```
              EACHNTH SAMPLING REPORT

                  INPUT PARAMETERS

    INPUT FILENAME                        PAYFILE
    INTERVAL SIZE                               5
    START RECORD                                4

                  SAMPLE FILE

    NUMBER OF RECORDS PROCESSED              1000
    NUMBER OF RECORDS SELECTED                 10

    FILE OUTFILE WILL BE CREATED
```

ENCRYPT

The ENCRYPT routine modifies an input file and creates an output file containing unintelligible data. A cryptographic technique randomly alters the contents of the data through the specification of a key value. This value becomes the key which locks and unlocks data in the file. To unlock data in the file, see the DECRYPT routine.

Note: The ENCRYPT and DECRYPT routines are not compatible across releases of CA-PanAudit Plus or CA-PanAudit.

Syntax

```
%ENCRYPT1 infile key  [LRECL length]

%ENCRYPT2 outfile [DBFILE infile] [STARTPOS identifier] [LENGTH value]
```

`infile`

Specify the name of the input file to ENCRYPT. A valid name is any previously defined file.

`key`

Specify an arbitrary number to initiate the pseudo random number generator that encrypts the file. This value becomes the key to decrypt the file and must be specified when the file is decrypted. Valid values include an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits long with no decimal places. Values greater than seven digits are truncated on the left.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 1 work byte + 4 RDW bytes = LRECL
```

`outfile`

Specify the name of the output file to which encrypted records are written. You must code file characteristics on the FILE statement for this output file. This output file must have the same file characteristics as the input file or have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

`[DBFILE infile]`

This is an optional parameter. You must specify this parameter only for database use of ENCRYPT. Specify the name of the input file to ENCRYPT. The name must be the same name that you specified for infile on the first invocation statement.

`[STARTPOS identifier]`

This optional parameter specifies the starting position for the encryption process. The identifier can be the name of a field or a numeric value.

If identifier is a field name, the encryption process starts at the first byte of the field. In this case, a valid value for identifier is any field defined in infile.

If identifier is a numeric value, the encryption process starts at the byte location that identifier specifies. For example, if identifier is the numeric value five, encryption begins with the fifth byte of the input file. In this case, a valid value for identifier is an actual numeric value only.

The default value for STARTPOS is the numeric value one, which means that the encryption begins with the first byte of infile. If you specify a value other than one for STARTPOS, then you must use LENGTH to avoid encrypting past the end of a record.

Encryption takes place starting at the file position that STARTPOS defines and continues for the length that you specified in the LENGTH parameter. If the default value is not used, you must specify the value for STARTPOS when the file is decrypted.

The requirements for the specification of STARTPOS are different when you use ENCRYPT in a database application. In this case the STARTPOS parameter is no longer optional—it is a required parameter. Furthermore, under these circumstances, STARTPOS cannot be a numeric value—it can only be a field name in a database record defined in the library section.

`[LENGTH value]`

This optional parameter specifies the length, or number of bytes, to be encrypted. The encryption process starts at the position that STARTPOS defines and continues for the length that the LENGTH parameter specifies. The default value for LENGTH is the record length of the current record. If the default value is not used, you must specify the value for LENGTH when the file is decrypted. A valid value is either an actual numeric value or the name of a field containing a numeric value. The value that you specify for LENGTH plus the numeric value of STARTPOS must be less than or equal to the length of the current record.

Operation — Stand-alone DISPLAY

STARTPOS and LENGTH are optional parameters. If the entire input file is to be encrypted, do not specify STARTPOS and LENGTH. They default to values which encrypt the input file from the first byte for the entire length of the input record.

This is extremely useful for variable length files. Because each record can have a different length, allowing these parameters to default ensures the encryption of all bytes of all records.

The value that you specify for LENGTH determines the number of encrypted bytes in the file. LENGTH does not determine the number of bytes written to the output file. ENCRYPT always creates records in the output file with the same record lengths found in the input file.

Since the encryption process renders the data unintelligible, it is important to remember the key, STARTPOS, and LENGTH values that ENCRYPT specifies (STARTPOS and LENGTH are required only if the default values are not used). The decryption process (DECRYPT) requires these values, and without them, it is difficult to decrypt the data and restore it for future processing. Carefully record and safeguard these values.

Operation — Database

The DBFILE parameter identifies ENCRYPT as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, when you are to use ENCRYPT in a database application, you must specify this parameter. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of ENCRYPT.

Note the difference in the specification of the STARTPOS parameter when you use ENCRYPT in a database application.

Also note that when ENCRYPT is used in a database application, even though it does not contain a REPORT statement, it has the limitations of a stand-alone REPORT routine.

Example

The following is an example of ENCRYPT.

Input

```
FILE INFILE  FB (200 10000)
  Field-name ...
  ...
FILE OUTFILE  FB (200 10000)
  Field-name ...
  ...
%ENCRYPT1 INFILE 139743
%ENCRYPT2 OUTFILE LENGTH 100
```

Results

This example demonstrates the encryption of the file INFILE starting at the first byte for a length of 100 bytes. The key value is 139743, and the encrypted file is written to the file OUTFILE.

EXPO

The EXPO routine calculates the result of raising a number to a specified power.

Syntax

```
%EXPO value exponent result [VALDEC dec1] [RESDEC dec2]
```

`value`

Specify the numeric value of the field being raised to the specified power. A valid value is an actual numeric value or the name of a field containing a numeric value. Values with decimal places more than defined by VALDEC are truncated to the right.

`exponent`

Specify the power to which the value parameter is raised. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can contain up to four decimal places and are truncated to the right if more than four are specified.

`result`

Specify the field to which the result of the calculation is placed. A valid name is any previously defined field. Fields with more decimal places defined in RESDEC are not accurate past the number of decimal places in RESDEC.

`[VALDEC dec1]`

Specify the number of decimal places needed in value. The default is 2. The maximum number of decimal places is 15.

`[RESDEC dec2]`

Specify the number of decimal places needed in result. The default is 4. The maximum number of decimal places is 15.

Operation — Inline

EXPO generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

An error condition can occur when the first parameter (value) is too large or the result is too large for the field definition. An error condition also occurs when the first parameter (value) is negative, and the second parameter (exponent) is not a whole number. When this occurs, the exponent is truncated to an integer, and the routine returns the value of YES in an internal field called EXPO-ERROR. The EXPO routine monitors these errors. The routine returns the value of YES in EXPO-ERROR for either error.

If you suspect that problems in the specification of the parameters may have occurred, check the field EXPO-ERROR after invoking the EXPO routine.

Operation — Database

No change in the specification of parameters is required to use EXPO with database files.

Example

The following is an example of EXPO.

Input

```
FILE ...
  NUMBER         1   6   N   2
  EXPONENT       7   6   N   4
  RESULT         W  10   N   4
  ...
JOB ...
%EXPO NUMBER EXPONENT RESULT
IF EXPO-ERROR EQ 'YES'
     DISPLAY 'ERROR DETECTED IN FOLLOWING RESULTS:'
END-IF
PRINT EXPO-REPORT
REPORT EXPO-REPORT
TITLE 1 'EXPONENTIATION LISTING'
LINE NUMBER EXPONENT RESULT
```

Output

In this example, the results of raising the field NUMBER to the power specified by EXPONENT are calculated. The field EXPO-ERROR is examined for the described error condition, and an appropriate message is printed in the report.

```
           EXPONENTIATION LISTING

    NUMBER      EXPONENT      RESULT

     20.10         3.2000   14,798.8301
     14.60         1.8000      124.6910
      3.40        3.0000       39.3040-
 ERROR DETECTED IN FOLLOWING RESULTS:
      3.40-        3.2000       39.3040-
       .         .          .
       .         .          .
       .         .          .
      4.80         5.2000    3,487.0200
```

Chapter 6

Generalized/Statistical Routines F-N

This chapter lists alphabetically, and gives detailed descriptions of, routines FILECOMP through NUMTEST.

FILECOMP

The FILECOMP routine compares specified locations in two files and produces a report of mismatched records. An optional key matching facility can be used to realign the files after a mismatch occurs.

Syntax

```
%FILECOMP primary secondary maximum {ALL                          }
                                    {'loc1,len1,loc2,len2,...,locx,lenx'} +

          [HEX  ]  [PRIKEYS 'pkey1 pkey2 ... pkey6']   +
          [NOHEX]
          [     ]
          [SECKEYS 'skey1 skey2 ... skey6']
```

`primary`

Specify the name of the primary input file. If record-matching keys are used, you must define the key fields specified in PRIKEYS in this file. A valid name is any previously defined file.

`secondary`

Specify the name of the secondary input file. If record-matching keys are used, you must define the key fields specified in SECKEYS in this file. A valid name is any previously defined file.

`maximum`

Specify the total number of record pairs that can be unequal before the file compare terminates. A valid value for maximum is an actual numeric value or the name of a field containing a numeric value.

```
{ALL                             }
{'loc1,len1,loc2,len2,...,locx,lenx'}
```

This parameter describes the locations in the records in the primary and secondary file that will participate in the file comparison.

ALL—Specifies that the full length of each record is to be compared, beginning with position 1.

loc1, len1, loc2, len2, . . .,locx, lenx—Specifies that certain portions of each record are to be compared.

The areas are defined using couplets:

- The first number designates the relative location in the record.
- The second number designates the length for which the compare is to be performed.

The entire string of numbers is enclosed in single quotes and entries are separated by commas. No spaces are allowed.

Note: You do not have to specify the couplets in ascending order according to location. For example, it may be more efficient to compare locations 74 in each record pair first, and then location 6. The compare of a record pair terminates, and records are selected for printing as soon as a mismatch in any position being compared is detected.

```
[HEX  ]
[NOHEX]
```

This optional parameter specifies whether you want a hexadecimal printout of the mismatched records included in the report.

HEX—A hexadecimal printout of mismatched records is included in the report. HEX is the default value.

NOHEX—The hexadecimal printing of records is eliminated from the report.

```
[PRIKEYS 'pkey1 pkey2 ... pkey6']
```

This optional parameter specifies the key fields used to realign the files when a mismatch occurs. You must define the fields pkey1 pkey2 ... pkey6 in the primary file. The parameters must be separated by a blank, and the entire string of key fields must be enclosed in single quotation marks.

The number of key fields specified in PRIKEYS must equal the number of key fields specified in SECKEYS. Six is the maximum number of key fields. A valid name is any alphanumeric or numeric field defined in the primary input file.

```
[SECKEYS 'skey1 skey2 ... skey6']
```

This optional parameter specifies the key fields used to realign the files when a mismatch occurs. You must define the fields skey1 skey2 ... skey6 in the secondary file. The parameters must be separated by a blank, and the entire string of key fields must be enclosed in single quotation marks.

The number of key fields specified in SECKEYS must equal the number of key fields specified in PRIKEYS. The maximum number of key fields is six. A valid name is any alphanumeric or numeric field defined in the secondary input file.

Operation — Stand-alone DISPLAY

Key fields defined in PRIKEYS and SECKEYS must be defined in the primary and secondary files. If the number of key fields defined in PRIKEYS and SECKEYS is not equal, an error message is printed, and execution stops. If PRIKEYS and SECKEYS are not specified, no file realignment is attempted. This means that if a mismatch occurs because of an unequal record pair, the remainder of the file may be interpreted as being mismatched, unless records become realigned by chance.

Operation — Database

FILECOMP cannot be used in a database application.

Examples

The following are two examples of the FILECOMP routine.

Example One

This example demonstrates a file comparison of all fields in the input files. No keys are used, so the files are not automatically realigned if they become offset by the addition or deletion of a record from either file.

Input

```
FILE INPUT1 ...
  Field-name ...
  ...
FILE INPUT2 ...
  Field-name ...
  ...
%FILECOMP INPUT1 INPUT2 5 ALL
```

Output

```
***** PAP220 - *UNEQUAL PAIR*                1  IN POSITION 17
    RECORD NUMBER            4  FROM INPUT1
CHAR CUSTOMER000000004        121083        00016
ZONE CEEEDDCDFFFFFFFFF00000200FFFFFF00000270FFFFF
NUMR 34236459000000000400001300C1210830000270C00016
     1...5...10...15...20...25...30...35...40...4

    RECORD NUMBER            4  FROM INPUT2
CHAR CUSTOMER000000007        121083        00016
ZONE CEEEDDCDFFFFFFFFF00000200FFFFFF00000270FFFFF
NUMR 3423645900000000070000130C1210830000270C00016
     1...5...10...15...20...25...30...35...40...4


***** PAP220 - *UNEQUAL PAIR*                2  IN POSITION 17
    RECORD NUMBER            8  FROM INPUT1
CHAR CUSTOMER000000008      - 040184        00024
ZONE CEEEDDCDFFFFFFFFF00000060FFFFFF00000490FFFFF
NUMR 3423645900000000080000810C0401840000590C00024
     1...5...10...15...20...25...30...35...40...4

    RECORD NUMBER            8  FROM INPUT2
CHAR CUSTOMER000000003      - 040184        00024
ZONE CEEEDDCDFFFFFFFFF00000060FFFFFF00000490FFFFF
NUMR 3423645900000000030000810C0401840000590C00024
     1...5...10...15...20...25...30...35...40...4


***** PAP220 - *UNEQUAL PAIR*                3  IN POSITION 16
    RECORD NUMBER           11  FROM INPUT1
CHAR CUSTOMER000000011     /  111883        00030
ZONE CEEEDDCDFFFFFFFFF00000680FFFFFF00000670FFFFF
NUMR 3423645900000000110000010C1118830000420C00030
     1...5...10...15...20...25...30...35...40...4

    RECORD NUMBER           11  FROM INPUT2
CHAR CUSTOMER000000021     /  111883        00030
ZONE CEEEDDCDFFFFFFFFF00000680FFFFFF00000670FFFFF
NUMR 3423645900000000210000010C1118830000420C00030
     1...5...10...15...20...25...30...35...40...4

***** PAP212 - END OF FILE REACHED ON INPUT1
***** PAP213 - END OF FILE REACHED ON INPUT2
***** PAP216 - FILE COMPARE ENDED
                   TOTAL OF            14 INPUT1 RECORDS READ
                   TOTAL OF            14 INPUT2 RECORDS READ
***** PAP218 - TOTAL OF                 3 UNEQUAL RECORDS FOUND
```

For each nonmatching pair of records, an informational message lists the accumulated number of nonmatching records, the position of the mismatching information, the record number and file name for the primary and secondary files, and a hexadecimal listing of each record. After all records are listed, other messages indicate when the end of file was encountered for both files, whether the compare ended normally or the count of records exceeded the maximum parameter, the total number of records in each file, and the total number of unequal records.

Example Two

This example demonstrates a file comparison of specified fields with the use of keys for realigning the files after an unequal pair of records is found. This example is similar to Example One, except that the files are realigned after a mismatch occurs. A message is printed indicating that end of file was first reached on INPUT1. The remainder of the records are unmatched.

Input

```
FILE INPUT1 ...
  IN1-DEPNO      13   5   N
  IN1-NAME        1  12   A
  ...
FILE INPUT2
  IN2-DEPNO      13   5   N
  IN2-NAME        1  12   A
  ...
%FILECOMP INPUT1 INPUT2 10 '13,5,40,5' PRIKEYS 'IN1-DEPNO IN1-NAME'
         SECKEYS 'IN2-DEPNO IN2-NAME'
```

Output

```
***** PAP220 - *UNEQUAL PAIR*                1  IN KEY A
    RECORD NUMBER            5  FROM INPUT1
CHAR CUSTOMER000000004        020484        00018
ZONE CEEEDDCDFFFFFFFFF00000170FFFFFF00000790FFFFF
NUMR 34236459000000004000070OC0204840000600C00018
     1...5...10...15...20...25...30...35...40...4

    RECORD NUMBER            5  FROM INPUT2
CHAR CUSTOMER000000005        020484        00018
ZONE CEEEDDCDFFFFFFFFF00000170FFFFFF00000790FFFFF
NUMR 34236459000000005000070OC0204840000600C00018
     1...5...10...15...20...25...30...35...40...4


***** PAP220 - *UNEQUAL PAIR*                2  IN KEY A
    RECORD NUMBER            6  FROM INPUT1
CHAR CUSTOMER000000008        040584     R  00020
ZONE CEEEDDCDFFFFFFFFF00000000FFFFFF00000970FFFFF
NUMR 34236459000000008000031OC0405840000990C00020
     1...5...10...15...20...25...30...35...40...4

    RECORD NUMBER            5  FROM INPUT2
CHAR CUSTOMER000000005        020484        00018
ZONE CEEEDDCDFFFFFFFFF00000170FFFFFF00000790FFFFF
NUMR 34236459000000005000070OC0204840000600C00018
     1...5...10...15...20...25...30...35...40...4


***** PAP212 - END OF FILE REACHED ON INPUT1
***** PAP214 - ALL REMAINING RECORDS ARE UNMATCHED
CHAR CUSTOMER000000014      & 030784        00032
ZONE CEEEDDCDFFFFFFFFF00000150FFFFFF00000700FFFFF
NUMR 34236459000000014000098OC0307840000770C00032
     1...5...10...15...20...25...30...35...40...4

CHAR CUSTOMER000000015        022084      - 00034
ZONE CEEEDDCDFFFFFFFFF00000430FFFFFF00000760FFFFF
NUMR 34236459000000015000049OC0220840000790C00034
     1...5...10...15...20...25...30...35...40...4

***** PAP216 - FILE COMPARE ENDED
                   TOTAL OF           11 INPUT1 RECORDS READ
                   TOTAL OF           13 INPUT2 RECORDS READ
***** PAP218 - TOTAL OF                4 UNEQUAL RECORDS FOUND
```

FILEGEN

The FILEGEN routine specifies the output file to which data, created by data generation routines, is written.

Syntax

```
%FILEGEN file number seed {HEX  }
                          {NOHEX}
```

`file`

Specify the name of the output file to which generated data is to be written. It corresponds to the name on the FILE statement in the library section of your program. A valid name is any previously defined file.

`number`

Specify the number of output records for which data is generated. Execution stops when the specified number of records have been generated. Valid values include an actual numeric value greater than 0 and less than 100,000,000.

`seed`

Specify an arbitrary number that initiates the internal random number generator for each data generation routine. If a different seed is specified in a rerun of an otherwise identical job, different data is generated. If the same seed is specified in a rerun of a job, the same data is generated. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits will be truncated on the left.

```
{HEX  }
{NOHEX}
```

Specify whether a hexadecimal listing of generated data is produced.

HEX—A hexadecimal listing of generated data is produced.

NOHEX—No listing is produced.

Operation — Inline

FILEGEN is invoked only once and must precede the data generation routines ALPHAGEN, DATEGEN, NUMGEN, or BADGEN. There is no input file to FILEGEN, so screening of input data is not allowed.

Operation — Database

FILEGEN and the associated routines ALPHAGEN, DATEGEN, NUMGEN, and BADGEN cannot be used in a database application.

Example

The following example illustrates the use of FILEGEN. An example of its use with the data generation routines is given following the discussion of the BADGEN routine.

Input

```
FILE PAYFILE ...
  Field-name ...
JOB INPUT NULL
  ...
%FILEGEN PAYFILE 500 17823 NOHEX
  ...
```

This example establishes the file name as PAYFILE and specifies that 500 records will be created. The seed for use in data generation routines is 17823, and a hexadecimal listing of the output file is not printed.

If the HEX option is specified, the output from FILEGEN can consist of a hexadecimal listing of the records created with the data generation routines.

FLDVALR

The FLDVALR routine validates a field against a specified range of values. A report of valid or invalid records is produced. Optionally, an output file of valid or invalid records may also be produced. If the field contains valid data, the field FLDVAL-SELECTED is set to VALID. If the field contains invalid data, the field FLDVAL-SELECTED is set to INVALID.

Syntax

```
%FLDVALR1 infile
%FLDVALR2 infile field {VALID  }  low  high       +
                       {INVALID}

               RPTSELECT {VALID  } {field2 field3} +
                         {INVALID} {SUMMARY      }

          [FILE outfile FILESELECT {VALID  }]
          [                        {INVALID}]
```

```
infile
```

Specify the name of the input file to FLDVALR. A valid name is any previously defined file.

```
field
```

Specify the name of the field to be validated. A valid name is any field defined in the input file.

```
{VALID  }
{INVALID}
```

Specify VALID if the range is to be considered a valid range. Specify INVALID if the range is to be considered an invalid range.

```
low
```

Low is the lower limit of the range of values. Low can be a literal value or the name of a field containing the value. If the field being validated is alphanumeric, the literal must be placed in triple quotes (‘’’literal’’’). The comparison between field and value is made under the rules of the CA-Easytrieve Plus Field Relational Condition. See the CA-Easytrieve Plus *Reference Guide* for complete rules.

```
high
```

High is the upper limit of the range of values. High can be a literal value or the name of a field containing the value. If the field being validated is alphanumeric, the literal must be placed in triple quotes (‘’’literal’’’). The comparison between field and value is made under the rules of the CA-Easytrieve Plus Field Relational Condition. See the CA-Easytrieve Plus *Reference Guide* for complete rules.

```
RPTSELECT {VALID  } {field2 field3}
          {INVALID} {SUMMARY      }
```

RPTSELECT defines the contents of the validation report.

VALID—When VALID is specified, the report will contain only valid records.

INVALID—When INVALID is specified, the report will contain only invalid records.

field2 field3—Specify field2 and field3 to request a detail report. This option lists three fields for each record in the file:

- Field (the field being validated)
- Field2
- Field3

Valid names are any previously defined field.

SUMMARY—Specify SUMMARY to request a report consisting of the total number of records selected for the input file.

```
[FILE outfile FILESELECT {VALID  }]
[                        {INVALID}]
```

This optional parameter specifies if an output file is to be created. Selected records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. A valid name for outfile is any previously defined file. FILESELECT specifies if the output file is to contain valid or invalid records.

VALID—When VALID is specified, the output file will contain only valid records.

INVALID—When INVALID is specified, the output file will contain only invalid records.

Operation — Stand-alone REPORT

FLDVALR reads the input file and determines whether the designated field contains valid or invalid databases upon the defined parameters.

FLDVALR will automatically produce a validation report as defined by the RPTSELECT parameters.

Optionally, an output file of valid or invalid records can also be produced by coding the FILE parameters.

Operation — Database

FLDVALR can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of FLDVALR.

Input

```
FILE PERSNL FB(150 1800)
  BRANCH     1  1 N
  REGION     2  2 N
  NAME      17 20 A
DEFINE LOW-VALUE  W  1 A  VALUE 'A'
DEFINE HIGH-VALUE W  1 A  VALUE 'F'
%FLDVALR1 PERSNL
%FLDVALR2 PERSNL NAME VALID LOW-VALUE HIGH-VALUE RPTSELECT VALID +
   REGION BRANCH
```

Output

```
                          FIELD VALIDATION REPORT
INPUT FILE: PERSNL        FIELD COMPARED: NAME        SELECT: VALID RANGE

     REPORT OF VALID RECORDS   NO OUTPUT FILE IS PRODUCED


                           NAME           REGION   BRANCH

                   BERG    NANCY            02       1
                   CORNING GEORGE           03       1
                   ARNOLD  LINDA            04       1
                   BRANDOW LYDIA            01       1
                   BYER    JULIE            04       1
                   DENNING RALPH            03       2
                   EPERT   LINDA            03       3
                   CROCI   JUDY             04       3
```

FLDVALT

The FLDVALT routine validates a field against a specified table of values. A report of valid or invalid records is produced. Optionally, an output file of valid or invalid records may also be produced. If the field contains valid data, the field FLDVAL-SELECTED is set to VALID. If the field contains invalid data, the field FLDVAL-SELECTED is set to INVALID.

Syntax

```
%FLDVALT1 infile
%FLDVALT2 infile field {VALID  }  table desclength +
                       {INVALID}

               RPTSELECT {VALID  } {field2 field3} +
                         {INVALID} {SUMMARY      }

          [FILE outfile FILESELECT {VALID  }]
          [                        {INVALID}]
```

`infile`

Specify the name of the input file to FLDVALT. A valid name is any previously defined file.

`field`

Specify the name of the field to be validated. A valid name is any field defined in the input file.

```
{VALID  }
{INVALID}
```

Specify VALID if the table contains valid values. Specify INVALID if the table contains invalid values.

```
table
```

Specify the name of the table to be used for validation. A valid name is any previously defined table.

```
desclength
```

Specify the length of the table's description field (used to define the length of FLDVALT-DESC). If a field is found to be valid, the table description is available in the field FLDVALT-DESC.

```
RPTSELECT {VALID  } {field2 field3}
          {INVALID} {SUMMARY      }
```

RPTSELECT defines the contents of the validation report.

VALID—When VALID is specified, the report will contain only valid records.

INVALID—When INVALID is specified, the report will contain only invalid records.

field2 field3—Specify field2 and field3 to request a detail report. This option lists three fields for each record in the file:

- Field (the field being validated)
- Field2
- Field3

Valid names are any previously defined field.

SUMMARY—Specify SUMMARY to request a report consisting of the total number of records selected for the input file.

```
[FILE outfile FILESELECT {VALID  }]
[                        {INVALID}]
```

This optional parameter specifies if an output file is to be created. Selected records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. A valid name for outfile is any previously defined file. FILESELECT specifies if the output file is to contain valid or invalid records.

VALID—When VALID is specified, the output file will contain only valid records.

INVALID—When INVALID is specified, the output file will contain only invalid records.

Operation — Stand-alone REPORT

FLDVALT reads the input file and determines whether the designated field contains valid or invalid data based upon a table lookup.

The FLDVALT routine searches for a match to any entry in the table specified in the table parameter. This table must be defined in a FILE statement, using the TABLE keyword. The table is either instream or external (a data file). For example:

```
FILE VALTTBL TABLE INSTREAM
ARG   1   2   N
DESC  3   8   A
01REGION 1
02REGION 2
03REGION 3
ENDTABLE
```

See the CA-Easytrieve Plus *Reference Guide* for complete information on table processing.

If a field is found to be valid (FLDVAL-SELECTED is VALID), the table description (for example, REGION 1) is moved to the field FLDVALT-DESC.

FLDVALT will automatically produce a validation report as defined by the RPTSELECT parameters.

Optionally, an output file of valid or invalid records can also be produced by coding the FILE parameters.

Operation — Database

FLDVALT can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of FLDVALT.

Input

```
FILE INFILE F(80)
  DATE    1  6 N  HEADING ('GREGORIAN' 'DATE') MASK 'Z9/99/99'
  MONTH   1  2 N
  DAY     3  2 N
  YEAR    5  2 N
FILE VALTBL TABLE INSTREAM
  ARG     1  2 N
  DESC    3 10 A
01 JANUARY
02 FEBRUARY
03 MARCH
10 OCTOBER
11 NOVEMBER
12 DECEMBER
ENDTABLE
*
%FLDVALT1 INFILE
%FLDVALT2 INFILE MONTH VALID VALTBL 10 RPTSELECT VALID +
    FLDVALT-DESC DATE
```

Output

```
                          FIELD VALIDATION REPORT
INPUT FILE: INFILE        FIELD COMPARED: MONTH        SELECT: VALID TABLE

     REPORT OF VALID RECORDS    NO OUTPUT FILE IS PRODUCED

                                             GREGORIAN
                      MONTH   FLDVALT-DESC     DATE

                       01       JANUARY       1/01/89
                       02       FEBRUARY      2/02/89
                       03       MARCH         3/03/89
                       01       JANUARY       1/02/89
                       12       DECEMBER     12/31/89
                       11       NOVEMBER     11/23/89
```

FLDVALV

The FLDVALV routine validates a field against a specified value.

A report of valid or invalid records is produced. Optionally, an output file of valid or invalid records may also be produced. If the field contains valid data, the field FLDVAL-SELECTED is set to VALID. If the field contains invalid data, the field FLDVAL-SELECTED is set to INVALID.

Syntax

```
%FLDVALV1 infile
%FLDVALV2 infile field {VALID  }  value           +
                       {INVALID}

               RPTSELECT {VALID  } {field2 field3} +
                         {INVALID} {SUMMARY      }

          [FILE outfile FILESELECT {VALID  }]
          [                        {INVALID}]
```

`infile`

Specify the name of the input file to FLDVALV. A valid name is any previously defined file.

`field`

Specify the name of the field to be validated. A valid name is any field defined in the input file.

```
{VALID  }
{INVALID}
```

Specify VALID if the range is to be considered a valid range. Specify INVALID if the range is to be considered an invalid range.

`value`

Value can be a literal value or the name of a field containing the value. If the field being validated is alphanumeric, the literal must be placed in triple quotes ('''literal'''). The comparison between field and value is made under the rules of the CA-Easytrieve Plus Field Relational Condition. See the CA-Easytrieve Plus *Reference Guide* for complete rules.

```
RPTSELECT {VALID  } {field2 field3}
          {INVALID} {SUMMARY      }
```

RPTSELECT defines the contents of the validation report.

VALID—When VALID is specified, the report will contain only valid records.

INVALID—When INVALID is specified, the report will contain only invalid records.

field2 field3—Specify field2 and field3 to request a detail report. This option lists three fields for each record in the file:

- Field (the field being validated)
- Field2
- Field3

Valid names are any previously defined field.

SUMMARY—Specify SUMMARY to request a report consisting of the total number of records selected for the input file.

```
[FILE outfile FILESELECT {VALID  }]
[                        {INVALID}]
```

This optional parameter specifies if an output file is to be created. Selected records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. A valid name for outfile is any previously defined file. FILESELECT specifies if the output file is to contain valid or invalid records.

VALID—When VALID is specified, the output file will contain only valid records.

INVALID—When INVALID is specified, the output file will contain only invalid records.

Operation — Stand-alone REPORT

FLDVALV reads the input file and determines whether the designated field contains valid or invalid databases upon the defined parameters.

FLDVALV will automatically produce a validation report as defined by the RPTSELECT parameters.

Optionally, an output file of valid or invalid records can also be produced by coding the FILE parameters.

Operation — Database

FLDVALV can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of FLDVALV.

Input

```
FILE PERSNL FB(150 1800)
  BRANCH     1  1 N
  REGION     2  2 N
  NAME      17 20 A
DEFINE TEST-VALUE W  1 N  VALUE 2
%FLDVALV1 PERSNL
%FLDVALV2 PERSNL REGION VALID TEST-VALUE RPTSELECT VALID +
   BRANCH NAME
```

Output

```
                           FIELD VALIDATION REPORT
INPUT FILE: PERSNL        FIELD COMPARED: REGION         SELECT: VALID VALUE

      REPORT OF VALID RECORDS   NO OUTPUT FILE IS PRODUCED


                    REGION   BRANCH         NAME

                      02       1      BERG     NANCY
                      02       1      NAGLE    MARY
                      02       2      POWELL   CAROL
                      02       2      KRUSE    MAX
                      02       3      THOMPSON JANICE
                      02       3      SMOTH    CINDY
                      02       3      ISAAC    RUTH
                      02       3      LACH     LORRIE
                      02       3      GRECO    LESLIE
                      02       3      REYNOLDS WILLIAM
                      02       4      JOHNSON  LISA
                      02       4      HAFER    ARTHUR
```

GAPCHCK

The GAPCHCK routine tests a numeric field to determine if any records in a continuous sequence are missing from the file. A report that lists any numbers missing from the sequence is automatically generated. For a series of missing numbers, the first and last numbers are printed. This listing provides information to verify missing items.

Syntax

```
%GAPCHCK1      infile     field  [LRECL length]

                {S}
%GAPCHCK2       {U}        [DBFILE infile]
```

`infile`

Specify the name of the input file to GAPCHCK. A valid name is any previously defined file.

`field`

Specify the name of the numeric field being tested.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 1 work byte + 4 RDW bytes = LRECL
```

```
{S}
{U}
```

Specify whether the records input to GAPCHCK are sorted or unsorted.

S—Indicates that the records are sorted in order by the field parameter. The sorted order must be in ascending sequence.

U—Indicates that records are not in sorted order. GAPCHCK sequences these records in temporary storage in ascending order by the field parameter before it begins the comparison process.

`[DBFILE infile]`

This is an optional parameter. You must specify this parameter only for database use of GAPCHCK. Specify the name of the input file to test. The name must be the same name that you specified for infile on the first invocation statement.

Operation — Stand-alone REPORT

GAPCHCK assumes the input file contains a continuous sequence of numbers in the designated field. You must specify whether the file is already in ascending sequence for the designated field. If necessary, GAPCHCK will sort the file. GAPCHCK then reads the file and determines whether the designated field contains any missing numbers.

GAPCHCK will automatically produce a report listing all missing numbers.

Operation — Database

The DBFILE parameter identifies GAPCHCK as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using GAPCHCK in a database application. Furthermore, you must specify all parameters on the second invocation statement, in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of GAPCHCK.

Example

This example uses GAPCHCK to sort an unsorted file and produce a report of missing numbers.

Input

```
FILE BUILD
CHECK#                  1    4    N
DESCRIPT                5   60    A
*
%GAPCHCK1 BUILD CHECK#
%GAPCHCK2 U
```

Output

```
6/25/90                  NUMERIC GAP TEST REPORT               PAGE      1


                       RANGE OF NUMBERS FOR    BUILD

                             0001    ---    0100



                             MISSING NUMBERS



                              0004

                              0010   ---   0011
                              0017   ---   0020
                              0032   ---   0034
                              0036
                              0038   ---   0040
                              0054   ---   0059
                              0061
                              0065
```

GETDATE

The GETDATE routine obtains the current date from the system and places it in the specified field. The resulting date contains no slashes, hyphens, or other non-numeric characters.

Syntax

```
%GETDATE field
```

`field`

Specify the name of the field to which the current date is placed. The date will be in the same format that is selected when the CA-PanAudit Plus system is installed. A valid field name is any previously defined field that will hold six characters.

Operation — Inline

GETDATE generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use GETDATE with database files.

Example

The following is an example of GETDATE.

Input

```
...
 DEFINE DATEFIELD    W   6   N
 JOB ...
 ...
 %GETDATE DATEFIELD
 ...
```

Results

GETDATE stores the current date in the field named DATEFIELD. The format of the value in DATEFIELD is defined by the option selected during CA-PanAudit Plus installation.

GETDATEL

The GETDATEL routine obtains the current date, in the CA-Easytrieve Plus SYSDATE-LONG format, and places it in the specified field. The resulting date contains no slashes, hyphens, or other non-numeric characters.

Note: GETDATEL functionality replaces GETDATE and supports a wider range of dates accurately. We recommend that systems running CA-Easytrieve Plus 6.2 or above use GETDATEL instead of GETDATE. GETDATEL is supported only in CA-Easytrieve Plus 6.2 and above.

Syntax

```
%GETDATEL field
```

`field`

Specify the name of the field to which the current date is placed. The date will be in the same format as SYSDATE-LONG. A valid field name is any previously defined field that will hold eight characters.

Example

The following is an example of GETDATEL.

Input

```
...
 DEFINE DATEFIELD    W   8   N
 JOB ...
 ...
 %GETDATEL DATEFIELD
 ...
```

Results

GETDATEL stores the current date in the field named DATEFIELD. The format of the value in DATEFIELD is defined by the option selected during CA-Easytrieve Plus 6.2 (or above) installation.

INTERVL

The INTERVL routine creates a frequency analysis on quantitative fields in a file. A report is produced, with an optional linear graph. The analysis performs the following functions automatically:

- Stratification of the file based on a specified interval size or a logarithmic interval specified in the optional parameter LOGARITHM
- Determination of low and high values
- Footing of the file
- Calculation of the mean and standard deviation for each stratum and for the entire file
- Printing of a graph based on the totals reported for each stratum
- Automatic placement in the first stratum of all records whose values are less than zero (unless stratified separately by an optional parameter)
- Automatic placement in the second stratum of all records whose values are equal to zero
- Automatic placement in the last stratum of all records whose values are greater than materiality

Syntax

```
[%INTERTAB number1 ... number255]

%INTERVL1 infile field {size     } {materiality} {GRAPH  {percent asterisks}}  +
                       {LOGARITHM} {INTERTAB   } {NOGRAPH                   }
                       {INTERTAB }

                        [           {ABS}]
       [LOWERLIM limit] [PERCENTAGE{POS}]   [LRECL length]
                        [           {NEG}]

%INTERVL2  [DBFILE infile] [REPORT {SHORT}]
                           [       {LONG }]
```

`[number1 ... number255]`

Specify actual numeric values which will be used as multiple defined end points. Required when INTERTAB is specified as the interval size and materiality. There is a maximum of 255 end points.

`infile`

Specify the name of the input file to INTERVL. A valid name is any previously defined file.

```
field
```

Specify the name of the quantitative field to be analyzed. A valid name is any field defined in the input file.

```
{size     }
{LOGARITHM}
{INTERTAB }
```

Define the interval size for data in the report. Valid values are:

- An actual numeric value
- The keyword LOGARITHM
- The keyword INTERTAB

size—Specify an actual numeric value. For example, if 1000 is specified, class intervals for the report are .01 to 1000.00, 1000.01 to 2000.00, and so on, up to the value specified in the materiality parameter.

LOGARITHM—Specify this keyword if logarithmic intervals are desired. For example, if LOGARITHM is specified, class intervals for the report are .01 to 1.00, 1.01 to 2.00,...9.01 to 10.0, 10.01 to 20.00,...90.01 to 100.00, and so on, up to the value specified in the materiality parameter.

INTERTAB—Specify this keyword if multiple defined end points are desired. The optional routine %INTERTAB must be coded if INTERTAB is specified.

If a value for LOWERLIM is specified, negative class intervals of this size are defined up to the value of LOWERLIM.

```
{materiality}
{INTERTAB   }
```

Specify a value that determines whether items are selected for frequency analysis or placed in a separate stratum of all items greater than materiality. If an input value is less than or equal to materiality, it participates in the frequency analysis. If it is greater than materiality, it is placed in the last stratum. Materiality is an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

Specify INTERTAB if multiple defined end points are desired. The optional routine %INTERTAB must be coded if INTERTAB is specified. The last end point specified for %INTERTAB is the materiality.

```
{GRAPH  }
{NOGRAPH}
```

Specify whether you want a graph to be produced with the report. This graph is a simple linear representation of the percentages of amounts in each interval. See Examples following the discussion of the INTERVL routine.

GRAPH—Specifies that a graph is to be produced. When this option is selected, the percent and asterisks parameters must be coded.

NOGRAPH—Specifies that the graph is to be omitted. When this option is selected, do not code the next two parameters.

```
{percent}
```

If GRAPH is specified, use this parameter to define the occurrence percentage at which graphing begins. The percentage value is subtracted from positive graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become less than zero are equal to zero. The percentage value is added to negative graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become greater than zero are set equal to zero.

For example, if percent is 10, graphing percentages from –10 percent to +10 percent are displayed with no asterisks. A standard graph contains no adjustment and is produced by specifying a percentage value of zero. A valid value for percentage is an actual integer value greater than or equal to zero and less than 100, or the name of a field containing an integer value in the same range. Values with decimal places are truncated on the right.

```
{asterisks}
```

In CA-PanAudit Plus, graph lines are drawn with the asterisk character. If GRAPH is specified, the value specified for the asterisks parameter defines the number of asterisks written to represent each percentage point. Valid values for asterisk include the actual numeric values 1 through 9 or the name of a field containing the values 1 through 9. For example, if an extremely flat distribution is anticipated, specify a higher value. This will make the graph easier to read. If a wide variance of percentages is anticipated, specify a lower value.

```
[LOWERLIM limit]
```

This optional parameter specifies stratification of the file for values less than zero. The value of limit specifies the lower limit of the stratification. When limit is not specified, INTERVL automatically stratifies all values less than zero into one stratum. When limit is specified, values are stratified in increments of the size parameter.

For example, if size is 1000, and limit is -3000, values less than zero are stratified into intervals of less than -3000.00, -3000.00 to -2000.01, -2000.00 to -1000.01, and -1000.00 to -.01 (see Example Two through Example Five). If limit is not specified, all values are stratified into one interval of values less than 0 (see Example One). A valid value for limit is an actual numeric value less than zero or the name of a field containing a numeric value less than zero.

```
[PERCENTAGE{ABS}]
[          {POS}]
[          {NEG}]
```

This optional parameter controls the method of calculating percentages when the file contains values less than zero. The default value is ABS. Example Two through Example Five demonstrate the use of this parameter.

ABS—Specifies that the absolute value of interval totals are used in calculating percentages. This method puts equal emphasis on both positive and negative values.

POS—Specifies that only positive interval totals are used in calculating percentages. Values less than zero are ignored for the purpose of percentages and graphing. This method places emphasis on positive values.

NEG—Specifies that negative interval totals are calculated with negative percentages. This method places emphasis on negative values. Negative percentages are printed with the character N; positive percentages are printed with the character *.

```
[LRECL length]
```

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 1 work byte + 4 RDW bytes = LRECL
```

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with INTERVL. INFILE identifies the name of the input file to INTERVL. The name must be the same name that is specified for INFILE on the first invocation statement.

```
[REPORT {SHORT}]
[       {LONG }]
```

This optional parameter specifies that a frequency analysis report is to be produced.

SHORT—Specifies that only a frequency analysis report of the designated field will be produced.

LONG—Specifies that a frequency analysis report of the designated field will be produced. Also, a report of the mean and standard deviation for each stratum and the entire file will be produced.

Operation — Stand-alone REPORT

Using the INTERVL routine to provide a representation of the distribution of values in a file can be useful in making meaningful parameter selections for other CA-PanAudit Plus routines. Before executing any statistical routine, you may find it helpful to run an INTERVL analysis to check that the parameters you have selected contain meaningful values and that the file contains the number or range of values you expected.

The optional LOWERLIM parameter allows you to control the effect on the distribution of values less than zero:

- If specified, intervals are stratified into increments the same size as the positive strata.
- If not specified, all negative values are stratified into one interval of values less than zero.

When you use the percent and asterisk parameters to create a graph, you must specify values that produce a meaningful graph. Many factors enter into this evaluation, including the limits of the printer you are using. For example, if a required graphing percentage is 30 percent, an asterisk value of 5 produces a line of 150 asterisks, which overflows most print lines. In general:

- Where the percentage of values to be graphed is large, give the asterisks parameter a low value (1 or 2).
- If a wide variance of percentages exists, specify a low value for asterisks.
- For graphing a narrow range of percentages, increase the asterisk value (3 through 9) for easier interpretation of results.
- If the graph overflows the print line, the letter O is printed between the PCT column and the graph to indicate the overflow condition.

- The calculated percentages in the graph are the percent of the amounts in each interval, not the percent of items in each interval.
- Depending on the value of the PERCENTAGE parameter, different totals are used for calculating percentages. The report lists the actual file total and the negative, positive, and absolute value totals.

The INTERVL routine creates a CA-PanAudit Plus table. Depending on the input data, the default allocation of 256 table entries may be exceeded. Error message A008 informs you of this condition. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*.

To increase the allocation for table entries, the CA-Easytrieve Plus options table must be link edited with a new maximum value. For details, see the CA-Easytrieve Plus *Getting Started* guide.

Operation — Database

The DBFILE parameter identifies INTERVL as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when you use INTERVL in a database application.

Examples

The following are six examples of INTERVL. Example One is a standard INTERVL. Example Two through Example Six demonstrate the use of the LOWERLIM and PERCENTAGE parameters.

Example One

This example demonstrates the use of INTERVL without the LOWERLIM or PERCENTAGE parameters. It contains intervals of 1000 with a materiality of 9000. A graph is produced which starts at 0 percent with each asterisk representing 1 percent. Percentages are calculated using the absolute value total.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERVL1 PAYFILE GROSS 1000 9000 GRAPH 0 1
%INTERVL2
```

Output

```
                            FREQUENCY ANALYSIS                PAGE      1
                 INPUT FIELD GROSS                        INPUT FILE: PAYFILE
                    INTERVAL SIZE: 1000          MATERIALITY:        9,000.00
                    MAXIMUM VALUE:        9,448.18  MINIMUM VALUE:        37.34

--------  RANGE  --------          TOTAL        COUNT      PCT       MEAN      STD DEV

             <     0.00              .00           0       .0         .00         .00
             =     0.00              .00           0       .0         .00         .00
        0.01 -  1000.00         9,120.30          16      1.0      570.02      287.71
     1000.01 -  2000.00        41,394.89          28      4.4    1,478.39      269.27
     2000.01 -  3000.00        68,072.57          26      7.2    2,618.18      273.88
     3000.01 -  4000.00        37,163.77          11      4.0    3,378.52      242.74
     4000.01 -  5000.00        53,603.23          12      5.7    4,466.94      247.03
     5000.01 -  6000.00       102,657.32          19     10.9    5,403.02      315.47
     6000.01 -  7000.00       131,526.82          20     14.0    6,576.34      316.13
     7000.01 -  8000.00       157,203.86          21     16.7    7,485.90      297.64
     8000.01 -  9000.00       256,609.39          30     27.3    8,553.65      299.69
             >  9000.00        82,475.51           9      8.8    9,163.95      120.02

     FINAL TOTALS                  939,827.66         192    100.0      4,894.94
     2,861.08
     POSITIVE TOTAL                939,827.66
     NEGATIVE TOTAL                       .00
     ABSOLUTE VALUE TOTAL          939,827.66

          FREQUENCY ANALYSIS GRAPH

                <     0.00            .0
                =     0.00            .0
           0.01 -  1000.00           1.0     *
        1000.01 -  2000.00           4.4     ****
        2000.01 -  3000.00           7.2     *******
        3000.01 -  4000.00           4.0     ****
        4000.01 -  5000.00           5.7     ******
        5000.01 -  6000.00          10.9     ***********
        6000.01 -  7000.00          14.0     **************
        7000.01 -  8000.00          16.7     *****************
        8000.01 -  9000.00          27.3     ***************************
                >  9000.00           8.8     *********
```

Example Two

This example demonstrates the use of INTERVL with the LOWERLIM parameter. It is the result of an INTERVL analysis with intervals of 1000 and materiality of 7000. The LOWERLIM parameter is specified to stratify the file for values less than zero. The PERCENTAGE parameter defaults to ABS, and the percentages are calculated using the absolute value total.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERVL1 PAYFILE GROSS 1000 7000 GRAPH 0 1 LOWERLIM -2000
%INTERVL2
```

Output

```
                                   FREQUENCY ANALYSIS                          PAGE     1
                         INPUT FIELD GROSS                                 INPUT FILE: PAYFILE
                            INTERVAL SIZE: 1000             MATERIALITY:          7,000.00
                            MAXIMUM VALUE:      7,638.46  MINIMUM VALUE:        2,891.00-

--------  RANGE  --------           TOTAL       COUNT     PCT          MEAN          STD DEV

           < -2000.00         80,611.45-       32      8.0         2,519.11-       219.12
 2000.00   - -1000.01         45,104.87-       30      4.5         1,503.50-       255.47
 1000.00   -     -0.01        10,869.13-       20      1.1           543.46-       247.45
           =      0.00              .00        0       .0              .00            .00
      0.01 -   1000.00        14,052.28        26      1.4           540.47        284.03
   1000.01 -   2000.00        59,801.76        39      6.0         1,533.38        304.32
   2000.01 -   3000.00        87,203.64        35      8.7         2,491.53        289.91
   3000.01 -   4000.00        81,786.69        23      8.2         3,555.94        299.82
   4000.01 -   5000.00       123,042.82        27     12.3         4,557.14        251.02
   5000.01 -   6000.00       204,098.29        37     20.4         5,516.17        303.70
   6000.01 -   7000.00       142,117.03        22     14.2         6,459.87        238.78
           >   7000.00       153,906.78        21     15.4         7,328.89        164.20

FINAL TOTALS                 729,423.84       312    100.2         2,337.90      3,066.48
POSITIVE TOTAL               866,009.29
NEGATIVE TOTAL               136,585.45-
ABSOLUTE VALUE TOTAL       1,002,594.74

    FREQUENCY ANALYSIS GRAPH

              < -2000.00              8.0        ********
     2000.00  - -1000.01              4.5        *****
     1000.00  -     -0.01             1.1        *
              =      0.00              .0
        0.01 -   1000.00              1.4        *
     1000.01 -   2000.00              6.0        ******
     2000.01 -   3000.00              8.7        *********
     3000.01 -   4000.00              8.2        ********
     4000.01 -   5000.00             12.3        ************
     5000.01 -   6000.00             20.4        ********************
     6000.01 -   7000.00             14.2        **************
              >   7000.00            15.4        ***************
```

Example Three

This example demonstrates the use of INTERVL with the LOWERLIM and PERCENTAGE parameters. It is identical to Example Two, except that the PERCENTAGE parameter is set to POS. This causes negative intervals to be disregarded in percentage calculations.

Note: The value of .0 is reported for PCT with negative intervals.

Percentages are calculated using the positive total.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERVL1 PAYFILE GROSS 1000 7000 GRAPH 0 1 LOWERLIM -2000    - PERCENTAGE POS
%INTERVL2
```

Output

```
                                 FREQUENCY ANALYSIS                                    PAGE      1
                        INPUT FIELD GROSS                                 INPUT FILE: PAYFILE
                            INTERVAL SIZE: 1000              MATERIALITY:         7,000.00
                            MAXIMUM VALUE:        7,665.55  MINIMUM VALUE:       2,898.12-

--------  RANGE  --------         TOTAL             COUNT       PCT         MEAN          STD DEV

          < -2000.00           60,493.44-            25        .0       2,419.74-         277.27
 2000.00  - -1000.01           33,712.60-            23        .0       1,465.77-         304.99
 1000.00  -    -0.01           18,758.36-            34        .0         551.72-         298.46
          =     0.00                 .00              0        .0            .00             .00
     0.01 -  1000.00           15,356.68             30       1.6         511.89          265.66
  1000.01 -  2000.00           37,258.36             25       4.0       1,490.33          279.18
  2000.01 -  3000.00           68,969.14             27       7.4       2,554.41          297.83
  3000.01 -  4000.00           85,176.72             24       9.1       3,549.03          318.90
  4000.01 -  5000.00          164,719.53             37      17.6       4,451.88          303.40
  5000.01 -  6000.00          143,907.81             26      15.4       5,534.92          274.30
  6000.01 -  7000.00          230,151.46             35      24.6       6,575.76          268.10
>  7000.00           190,896.95            26      20.4       7,342.19          178.85

FINAL TOTALS                  823,472.25            312     100.1       2,639.33        3,123.59
POSITIVE TOTAL                936,436.65
NEGATIVE TOTAL                112,964.40-
ABSOLUTE VALUE TOTAL        1,049,401.05

    FREQUENCY ANALYSIS GRAPH

              < -2000.00              .0
     2000.00  - -1000.01              .0
     1000.00  -    -0.01              .0
              =     0.00              .0
         0.01 -  1000.00             1.6      **
      1000.01 -  2000.00             4.0      ****
      2000.01 -  3000.00             7.4      *******
      3000.01 -  4000.00             9.1      *********
      4000.01 -  5000.00            17.6      ******************
      5000.01 -  6000.00            15.4      ***************
      6000.01 -  7000.00            24.6      *************************
              >  7000.00            20.4      ********************
```

Example Four

This example demonstrates the use of INTERVL with the LOWERLIM and PERCENTAGE parameters. This report is identical to Example Two and Example Three except that the PERCENTAGE parameter is set to NEG. This causes negative intervals to be calculated as negative percentages. Percentages are calculated using the value for final totals.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERVL1 PAYFILE GROSS 1000 7000 GRAPH 0 1 LOWERLIM -2000    - PERCENTAGE NEG
%INTERVL2
```

Output

```
                                           FREQUENCY ANALYSIS                      PAGE      1
                         INPUT FIELD GROSS                                 INPUT FILE: PAYFILE
                             INTERVAL SIZE: 1000            MATERIALITY:          7,000.00
                            MAXIMUM VALUE:      7,665.55  MINIMUM VALUE:         2,898.12-

--------  RANGE  --------          TOTAL         COUNT      PCT         MEAN        STD DEV

             < -2000.00          60,493.44-       25       7.3-      2,419.74-      277.27
    -2000.00 - -1000.01          33,712.60-       23       4.1-      1,465.77-      304.99
    -1000.00 -     -0.01         18,758.36-       34       2.3-        551.72-      298.46
             =      0.00               .00         0        .0           .00          .00
        0.01 -   1000.00         15,356.68        30       1.9         511.89       265.66
     1000.01 -   2000.00         37,258.36        25       4.5       1,490.33       279.18
     2000.01 -   3000.00         68,969.14        27       8.4       2,554.41       297.83
     3000.01 -   4000.00         85,176.72        24      10.3       3,549.03       318.90
     4000.01 -   5000.00        164,719.53        37      20.0       4,451.88       303.40
     5000.01 -   6000.00        143,907.81        26      17.5       5,534.92       274.30
     6000.01 -   7000.00        230,151.46        35      27.9       6,575.76       268.10
             >   7000.00        190,896.95        26      23.2       7,342.19       178.85

FINAL TOTALS                    823,472.25       312     100.0       2,639.33      ,123.59
POSITIVE TOTAL                  936,436.65
NEGATIVE TOTAL                  112,964.40-
ABSOLUTE VALUE TOTAL          1,049,401.05

     FREQUENCY ANALYSIS GRAPH

               < -2000.00             7.3-     *******
      2000.00  - -1000.01             4.1-     ****
      1000.00  -     -0.01            2.3-     **
               =      0.00             .0
          0.01 -   1000.00            1.9      **
       1000.01 -   2000.00            4.5      *****
       2000.01 -   3000.00            8.4      ********
       3000.01 -   4000.00           10.3      **********
       4000.01 -   5000.00           20.0      ********************
       5000.01 -   6000.00           17.5      ******************
       6000.01 -   7000.00           27.9      ****************************
               >   7000.00           23.2      ***********************
```

Example Five

This example demonstrates the use of INTERVL with the LOGARITHM function. It uses the LOGARITHMIC stratification feature with a materiality of 9000.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERVL PAYFILE GROSS LOGARITHM 9000 NOGRAPH
%INTERVL2
```

Output

```
                                       FREQUENCY ANALYSIS                         PAGE      1
                       INPUT FIELD GROSS                                  INPUT FILE: PAYFILE
                           INTERVAL SIZE: LOGARITHM         MATERIALITY:          9,000.00
                          MAXIMUM VALUE:        7,665.55  MINIMUM VALUE:        2,898.12-

-------- RANGE  -------------     TOTAL        COUNT     PCT          MEAN            STD DEV

           <     0.00          112,964.40-       82     10.8        1,377.61-          838.20
           =     0.00                 .00         0       .0             .00             .00
       0.01 -    1.00                 .00         0       .0             .00             .00
       1.01 -    2.00                 .00         0       .0             .00             .00
       2.01 -    3.00                 .00         0       .0             .00             .00
       3.01 -    4.00                 .00         0       .0             .00             .00
       4.01 -    5.00                 .00         0       .0             .00             .00
       5.01 -    6.00                 .00         0       .0             .00             .00
       6.01 -    7.00                 .00         0       .0             .00             .00
       7.01 -    8.00                 .00         0       .0             .00             .00
       8.01 -    9.00                 .00         0       .0             .00             .00
       9.01 -   10.00                 .00         0       .0             .00             .00
      10.01 -   20.00               13.68         1       .0           13.68             .00
      20.01 -   30.00                 .00         0       .0             .00             .00
      30.01 -   40.00                 .00         0       .0             .00             .00
      40.01 -   50.00                 .00         0       .0             .00             .00
      50.01 -   60.00                 .00         0       .0             .00             .00
      60.01 -   70.00                 .00         0       .0             .00             .00
      70.01 -   80.00                 .00         0       .0             .00             .00
      80.01 -   90.00               83.29         1       .0           83.29             .00
      90.01 -  100.00                 .00         0       .0             .00             .00
     100.01 -  200.00              283.98         2       .0          141.99           38.04
     200.01 -  300.00            1,324.37         5       .1          264.87           31.99
     300.01 -  400.00            1,409.96         4       .1          352.49           18.66
     400.01 -  500.00                 .00         0       .0             .00             .00
     500.01 -  600.00            1,670.75         3       .2          556.92           31.71
     600.01 -  700.00            3,217.31         5       .3          643.46           23.16
     700.01 -  800.00            2,986.94         4       .3          746.74           30.47
     800.01 -  900.00            2,472.11         3       .2          824.04           12.20
     900.01 - 1000.00            1,894.29         2       .2          947.15           25.89
    1000.01 - 2000.00           37,258.36        25      3.6        1,490.33          279.18
    2000.01 - 3000.00           68,969.14        27      6.6        2,554.41          297.83
    3000.01 - 4000.00           85,176.72        24      8.1        3,549.03          318.90
    4000.01 - 5000.00          164,719.53        37     15.7        4,451.88          303.40
    5000.01 - 6000.00          143,907.81        26     13.7        5,534.92          274.30
    6000.01 - 7000.00          230,151.46        35     21.9        6,575.76          268.10
    7000.01 - 8000.00          190,896.95        26     18.2        7,342.19          178.85
    8000.01 - 9000.00                 .00         0       .0             .00             .00
           > 9000.00                  .00         0       .0             .00             .00

FINAL TOTALS                   823,472.25       312    100.0        2,639.33        3,123.59
POSITIVE TOTAL                 936,436.65
NEGATIVE TOTAL                 112,964.40-
ABSOLUTE VALUE TOTAL         1,049,401.05
```

Example Six

This example demonstrates the use of INTERVL with the INTERTAB parameter.

Input

```
FILE PAYFILE FB (44 4400)
GROSS 31 14 N 2
%INTERTAB 600 700 900 1100 1300
%INTERVL1 PAYFILE GROSS INTERTAB INTERTAB GRAPH 0 1
%INTERVL2
```

Output

```
                                         FREQUENCY ANALYSIS                        PAGE      1
                      INPUT FIELD GROSS                                    INPUT FILE: PAYFILE
                          INTERVAL SIZE: INTERTAB         MATERIALITY:           1,300.00
                         MAXIMUM VALUE:        7,665.55  MINIMUM VALUE:        2,898.12-

--------- RANGE  -------------       TOTAL         COUNT      PCT        MEAN           STD DEV

             <     0.00        112,964.40-         82      10.8       1,377.61-          838.20
             =     0.00               .00           0        .0            .00              .00
        0.01 -   600.00          4,786.03          16        .5         299.13           159.39
      600.01 -   700.00          3,217.31           5        .3         643.46            23.16
      700.01 -   900.00          5,459.05           7        .5         779.86            45.36
      900.01 -  1100.00          4,993.20           5        .5         998.64            46.48
     1100.01 -  1300.00          6,175.43           5        .6       1,235.09            32.92
             >  1300.00        911,805.63         192      86.9       4,748.99         1,844.69

FINAL TOTALS                   823,472.25         312     100.1       2,639.33         3,123.59
POSITIVE TOTAL                 936,436.65
NEGATIVE TOTAL                 112,964.40-
ABSOLUTE VALUE TOTAL          ,049,401.05

     FREQUENCY ANALYSIS GRAPH

               <     0.00            10.8      ***********
               =     0.00              .0
          0.01 -   600.00              .5      *
        600.01 -   700.00              .3
        700.01 -   900.00              .5      *
        900.01 -  1100.00              .5      *
       1100.01 -  1300.00              .6      *
               >  1300.00            86.9  (O)
```

INTSAMP

The INTSAMP routine creates a statistically valid random sample from an existing file based upon an interval of records. Each interval contains a given number of records specified by the size parameter. INTSAMP randomly selects one record from each interval and can optionally write it to an output file.

Syntax

```
%INTSAMP1 infile size seed

%INTSAMP2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to INTSAMP. A valid name is any previously defined file.

`size`

Specify the size of each interval. Each interval will contain the number of records specified by this parameter. One record is randomly selected from each interval. A valid value for size is an actual numeric value greater than or equal to one or the name of a field containing a numeric value greater than or equal to one.

`seed`

Specify an arbitrary number that initiates the random number generator. The seed determines which record is randomly selected from each interval. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left. Seed and size mutually affect record selection. For additional information, see Operation — Stand-alone DISPLAY in this routine.

```
{outfile}
{NOFILE }
```

Specify whether records selected in each interval are to be written to an output file.

outfile—Records selected are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records are not written to an output file.

`[DBFILE infile]`

This optional parameter specifies the database file for use with INTSAMP. Infile identifies the name of the input file to INTSAMP. The name must be the same name that you specified for infile on the first invocation statement.

`[PERFORM procname]`

Specify the name of a CA-Easytrieve Plus procedure that is performed by the INTSAMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field INTSAMP-SELECTED is set to the value YES. If a record is not selected for the sample file, INTSAMP-SELECTED is set to the value NO.

After the invocation of INTSAMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the INTSAMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that INTSAMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

A record is randomly selected from each interval by the RANDOM routine and written to the output file. If the last interval is not a full interval, INTSAMP may or may not write a record from that interval to the sample file, depending on how full the last interval is and the value of the seed parameter.

For example, if there are 1000 records in each interval and the last interval contains only 100 records, there is approximately a 10 percent chance that a record will be selected from this interval. The value of seed determines which record is written from this interval of 100 records. A seed value of 14583 may specify that the 785th record in this interval will be selected. Because there are only 100 records, no record is written to the sample file. On the other hand, a seed value of 842 may specify that the 56th record in the 100 record interval is to be selected. In this case, a record is written to the sample file.

The exact record selected for any given interval depends on the value of the seed, the number of records in each interval, and the actual interval which is being examined.

Operation — Database

The DBFILE parameter identifies INTSAMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using INTSAMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of INTSAMP.

Example

The following is an example of INTSAMP.

Input

```
FILE INFILE FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE FB (44 4400)
%INTSAMP1 INFILE 100 97
%INTSAMP2 OUTFILE
```

Output

The report shows that there were 12353 records processed from the input file. The interval size of 100 specifies that from each group of 100 records, one random sample is written to the output file. There are 123 full intervals of 100 records and one partial interval of 53 records. One random sample is selected from each full interval. A record is not selected from the partial interval of 53 records.

```
                    INTERVAL SAMPLING REPORT

                        INPUT PARAMETERS

          INPUT FILE NAME                         INFILE
          INTERVAL SIZE                             100
                        SAMPLE FILE

          NUMBER OF RECORDS IN INPUT FILE        12,353
          NUMBER OF RECORDS PROCESSED               123

          FILE OUTFILE WILL BE CREATED
```

MULTDUP

The MULTDUP routine compares specified fields within records in the input file to determine if duplicates exist. You can compare from one to 50 fields, referred to as keys. Records are selected as duplicates when all of the specified key fields match. Duplicate records can be written to an output file.

MULTDUP differs from DUPTEST in that it permits you to examine multiple fields for duplicates, instead of just one field.

Syntax

```
%MULTDUP1 infile  [LRECL length]

%MULTDUP2 infile {S} {outfile} key1 key2 ... keyn
                 {U} {NOFILE }
```

`infile`

Specify the name of the input file to MULTDUP. A valid name is any previously defined file.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 1 work byte + 4 RDW bytes = LRECL
```

`{S}`
`{U}`

Specify whether the records input to MULTDUP are sorted or unsorted.

S—Indicates that records are sorted in the order specified by the key fields. The sorted order can either be in ascending or descending sequence. For more details about sort requirements, see Operation — Stand-alone REPORT on the following page.

U—Indicates that records are not in sorted order. MULTDUP will sequence the records in temporary storage in the order specified by the key fields before it begins the comparison process.

```
{outfile}
{NOFILE }
```

Specify whether an output file of duplicate records is to be created.

outfile—Duplicate records are written to the output file indicated by this parameter. File characteristics must be coded on the FILE statement for this output file. Valid names for outfile include any previously defined file.

NOFILE—Duplicate records are not written to an output file.

```
key1 key2 ... keyn
```

List the fields that are to participate in the search for duplicate records. Records are selected as duplicates only when all of the specified key fields match. You can specify a maximum of 50 key fields. A valid key nonquantitative field is any field defined in the input file.

Operation — Stand-alone REPORT

Before MULTDUP can test for duplicates, input records must be sorted according to the comparison fields. For example, if two key fields are specified, records must be sorted by key1 and by the key2 field in key1. Key1 is referred to as the major sort field, and key2 the minor sort field. If sort indicator S is specified, MULTDUP assumes that the file is sorted. If sort indicator U is specified, MULTDUP sorts the records according to the key1 through key*n* parameters.

The MULTDUP routine does not produce a report. For each duplicate condition detected, an internal flag MULTDUP-FLAG is set to the value YES. If no duplicate condition exists, the MULTDUP-FLAG is set to the value NO.

To create a listing of duplicate records, code the appropriate IF and END-IF statements to test the MULTDUP-FLAG field immediately following the invocation of MULTDUP. When the flag is YES, the CA-Easytrieve Plus statements PRINT or DISPLAY can be used to create the desired listing (see the CA-Easytrieve Plus *Reference Guide).* Examples of report processing with MULTDUP can be found in the examples on the following page.

Even though MULTDUP does not contain a REPORT statement, it functions as a Stand-alone REPORT routine. This means that, even though a display statement is used for output (see Example One), MULTDUP still has all the limitations of a Stand-alone REPORT routine.

Operation — Database

MULTDUP can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can either be a nondatabase file name or the name of a database file defined in the library section.

Examples

The following two examples demonstrate the use of MULTDUP.

Example One

Input

```
FILE ACCOUNT
  FIRSTNM         1   8   A
  MIDDLE          9   1   A
  LAST           10  10   A
  ACCOUNT-NUM    20   7   N
  ...
%MULTDUP1 ACCOUNT
%MULTDUP2 ACCOUNT U NOFILE LAST FIRSTNM MIDDLE
IF MULTDUP-FLAG EQ 'YES'
     DISPLAY +5, ACCOUNTNUM, +5, FIRSTNM, +1, MIDDLE, +1, LAST
END-IF
...
```

Output

This example demonstrates the use of the DISPLAY statement to create a listing of duplicate records. When a duplicate record is detected, the MULTDUP-FLAG field is set to the value YES.

```
0283747         JOAN R      JONES
0343346         JOAN R      JONES
0745785        LARRY L     LITTLE
0748658        LARRY L     LITTLE
0798223        LARRY L     LITTLE
0103467         JOHN L WILLIAMSON
0187645         JOHN L WILLIAMSON
```

The IF statement tests for the YES condition and displays four fields if a duplicate record is found. The first field printed is an identification field, such as an account number, to assist in identifying the duplicate record. The remaining fields are the actual duplicate fields.

Example Two

Input

```
FILE ACCOUNT
  FIRSTNM         1   8   A
  MIDDLE          9   1   A
  LAST           10  10   A
  ACCOUNT-NUM    20   7   N
FILE DUPFILE
...
%MULTDUP1 ACCOUNT
%MULTDUP2 ACCOUNT S DUPFILE LAST FIRSTNM MIDDLE
IF MULTDUP-FLAG EQ 'YES'
     PRINT DUPLICATE-REPORT
END-IF
REPORT DUPLICATE-REPORT
TITLE 1  'DUPLICATE RECORDS ON THE CUSTOMER FILE'
HEADING  ACCOUNT-NUM  'ACCOUNT NUMBER'
HEADING FIRSTNM 'FIRST'
LINE ACCOUNT-NUM FIRSTNM MIDDLE LAST
```

Output

This example demonstrates the use of the PRINT statement to create a listing of duplicate records. When a duplicate record is located, the MULTDUP-FLAG field is set to the value YES.

The IF statement tests for the YES condition and executes the report named DUPLICATE-REPORT if a duplicate record is found. DUPLICATE-REPORT automatically prints page numbers, column headings, and the report title as specified. For information on report processing parameters, see the CA-Easytrieve Plus *Reference Guide.*

```
3/14/84      DUPLICATE RECORDS ON THE CUSTOMER FILE      PAGE   1

          ACCOUNT NUMBER   FIRST  MIDDLE      LAST

            0283747         JOAN    R          JONES
            0343346         JOAN    R          JONES
            0745785        LARRY    L         LITTLE
            0748658        LARRY    L         LITTLE
            0798223        LARRY    L         LITTLE
            0103467         JOHN    L     WILLIAMSON
            0187645         JOHN    L     WILLIAMSON
```

MULTREG

The MULTREG routine performs a multiple regression and correlation analysis. This routine solves the regression equation for two or three variables:

```
y = a + bx1 + cx2 +dx3
```

Syntax

```
%MULTREG1 infile type field1 field2 field3 [field4]
%MULTREG2
```

`infile`

Specify the name of the input file to MULTREG. A valid name is any previously defined file.

`type`

Specify the number of variables. Specify 2 for two variables or 3 for three independent variables. A valid value is the actual numeric value 2 or 3, or the name of a field containing a 2 or a 3.

`field1`

Specify the name of the field containing the y value in the previous equation. A valid value is any numeric field defined in the input file. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`field2`

Specify the name of the field containing the x1 value in the previous equation. A valid value is any numeric field defined in the input file. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`field3`

Specify the name of the field containing the x2 value in the previous equation. A valid value is any numeric field defined in the input file. Values may contain up to two decimal places and will be truncated on the right if more than two are specified.

`[field4]`

This optional parameter specifies the name of the field containing the x3 value in the previous equation. A valid value is any numeric field defined in the input file. Values can contain up to two decimal places and are truncated on the right if more than two are specified. This parameter must not be specified when type equals two.

Operation — Stand-alone DISPLAY

The variables to be analyzed must be contained on the input file with each record containing the corresponding y, x1, x2, and x3 values.

Operation — Database

MULTREG can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of MULTREG.

Input

```
FILE INFILE FB (20 2000)
X1VALUE  1 5 N
X2VALUE  6 5 N
X3VALUE 11 5 N
YVALUE  16 5 N
%MULTREG1 INFILE 3 YVALUE X1VALUE X2VALUE X3VALUE
%MULTREG2
```

Output

```
                                       MULTIPLE REGRESSION ANALYSIS

                                        3    INDEPENDENT VARIABLES

                                             INPUT PARAMETERS

                       INPUT FILE NAME                                INFILE
                        Y FIELD                                      YVALUE
                       X1 FIELD                                     X1VALUE
                       X2 FIELD                                     X2VALUE
                       X3 FIELD                                     X3VALUE

REGRESSION EQUATION ANALYSIS
                              COEFFICIENT                  MEAN                   STD DEV
CONSTANT                        229.10-
VARIABLE X1                       3.03-                    250.00                  112.37
VARIABLE X2                        .04-                   471.50                  141.42
VARIABLE X3                      13.44-                    33.75                   25.06
Y VALUE                                                    52.13                   29.38

MULTIPLE R                      .1807
R SQUARE                        .0327
STD ERROR OF EST                 29.35

ANALYSIS OF VARIANCE
                             DF           SUM OF SQUARES           MEAN SQ              F RATIO
REGRESSION                    3                 2,791.31            930.44               1.0804
RESIDUAL                     96                82,674.00            861.19

CORRELATION MATRIX
X2                       .8470
```

```
X3                        .9999                 .8463
Y                         .0744-                .1199-                .0756-
                             X1                    X2                    X3

NUMBER OF RECORDS                   100
```

The report presents an analysis of the effects of three independent variables on the dependent variable y. The analysis solves the regression equation, analyzes the variance, and presents the correlation matrix.

The report lists the coefficients, mean, and standard deviation of the variables x1, x2, and x3. It also lists the a value: the constant in the previous equation.

Statistics are presented which demonstrate the relationship of the variables to the regression line. The multiple R value and the corresponding R SQUARE are an expression of the correlation between the regression line and the actual x1, x2, x3, and y values. Values for R range between -1.0 and +1.0, with 1.0 being maximum inverse correlation and +1.0 maximum positive correlation.

R SQUARE is another measurement of correlation, called the Coefficient of Determination, whose values range between 0 and +1.0. It is a more meaningful indicator of the relationship between y and x than is the R value.

The analysis of variance topic describes the degrees of freedom, sum of squares of deviation, mean squared deviation, and the F ratio. These are various measurements of the accuracy of the regression line in relation to the actual input values.

NUMGEN

The NUMGEN routine generates numeric data and writes it to the field parameter.

Note: The FILEGEN routine must be invoked use the NUMGEN routine.

Syntax

```
                {BETWEEN    minimum maximum             }
%NUMGEN field   {SEQUENCE   from  to  increment         }
                {CONSTANT   'value1,value2,...,valuen'  }
```

`field`

Specify the name of the field where NUMGEN places the numeric data it generates. Valid names include any numeric field, defined in the file name specified in FILEGEN, with a data format of N (zoned decimal), P (packed decimal), B (binary), or U (unsigned packed decimal). The maximum number of generated digits for field is 15.

`{BETWEEN minimum maximum}`

The BETWEEN keyword causes random numbers to be generated for the field you specified in the field parameter. Generated values will be in the range you specify by the minimum and maximum parameters. Valid values for minimum and maximum are either actual numeric values or the name of a field containing the value. Minimum and maximum must be less than 100 trillion and may contain up to four decimal places. The seed for the random generation of values is obtained from the FILEGEN routine.

Values generated are greater than or equal to the minimum and less than the maximum. If the desired range of values is 10.75 through 93.62, specify a minimum of 10.75 and a maximum of 93.63. For integer values, if the desired range is 1 through 10, specify a minimum of 1 and a maximum of 11.

`{SEQUENCE from to increment}`

The SEQUENCE keyword causes a fixed set of numbers to be generated for the field you specified in the field parameter. The set of numbers begins with the from value and is incremented for each record by the increment value until the to value is equaled or exceeded. The sequence is repeated beginning with the from value. Valid values for from, to, and increment are actual numeric values or the name of a field containing the value.

To generate a decreasing set of numbers, specify a from value greater than the to value, and code a negative increment.

`{CONSTANT 'value1,value2,. . .,valuen'}`

The CONSTANT keyword causes a specified series of values to be generated for the field you specified in the field parameter. The sequence is repeated until a value has been generated for each record being created.

Separate the values in the CONSTANT string by commas, and enclose the string in single quotation marks. Valid values consist of actual numeric values only. The entire length of the literal portion of CONSTANT is limited to 40 characters, including commas.

Operation — Inline

Use the NUMGEN routine only after you have specified the FILEGEN routine. NUMGEN can be specified with ALPHAGEN, DATEGEN, and BADGEN. Conditional execution of NUMGEN is discussed in Conditional Execution of Data Generation Routines following the BADGEN routine.

Operation — Database

NUMGEN, with FILEGEN, cannot be used in a database application.

Examples

The following example illustrates how NUMGEN is used with FILEGEN:

```
FILE CENTFILE F(24)
  NAME    1 12 A
  EMP#   13  5 N
  BIRTH  18  6 N
*
JOB INPUT NULL
  %FILEGEN CENTFILE 25 5 NOHEX
  %ALPHAGEN NAME ‘ CUST ‘ CONSTANT ‘JOHNSON,SMITH,PETERS’
  %NUMGEN EMP# SEQUENCE 1 10050 1
  %DATEGEN BIRTH MMDDYY 0  BETWEEN 010151 010175
```

The following examples demonstrate the three uses of NUMGEN. An example of the full facilities of the test data generation routines is found following the BADGEN routine.

Between

This example generates a random value between 3 and 54 for each record being created and writes it to the ZONE field.

```
%NUMGEN ZONE BETWEEN 3 55
SEQUENCE
```

A value of 80 is written in the DEPT field of the first record, 75 to the second record, 70 to the third record, and so on, until the value 10 is reached. This sequence is repeated beginning with the value 80.

```
%NUMGEN DEPT SEQUENCE 80 10 -5
```

Constant

The value 1 is written in the CODE field of the first record, 3 to the second record, 5 to the third record, and so on, until the value 8 is reached. This sequence is repeated beginning with the value 1.

```
%NUMGEN CODE CONSTANT '1,3,5,7,9,2,4,6,8'
```

NUMTEST

The NUMTEST routine evaluates the contents of a specified field for valid numeric data. For each record with non-numeric data in the specified field, an identifying field with a description is printed followed by the hexadecimal representation of the non-numeric field. If the field contains valid numeric data, the field NUMTEST-FLAG is set to the value YES. If the field does not contain valid numeric data, the NUMTEST-FLAG is set to the value NO.

Syntax

```
%NUMTEST field 'descrip' idfield
```

`field`

Specify the name of the field being tested. Valid names include any previously defined field.

`'descrip'`

Code a literal description. This is printed for each non-numeric field value encountered. The maximum length of this parameter is limited by the maximum length of a report line defined at installation. As long as the description is enclosed in single quotation marks, there is no restriction on its content. If you do not enclose the description in quotes, it must not contain any blank characters.

`idfield`

Specify the name of the ID field. This field is printed on the listing to identify the record with the non-numeric value. Valid names include any previously defined field. Probable choices for the ID field might be account number or invoice number.

Operation — Inline

For each non-numeric value of field, NUMTEST prints the ID field followed by the description and the hexadecimal representation of the field. If the field contains valid numeric data, an internal field NUMTEST-FLAG is set to the value YES. If the field does not contain valid numeric data, NUMTEST-FLAG is set to the value NO.

Operation — Database

No change in the specification of parameters is required to use NUMTEST with database files.

Example

The following is an example of NUMTEST.

Input

```
FILE ...
  ACTNO         1   5   N
  BALANCE       6   7   N   2
  ...
JOB ...
%NUMTEST BALANCE 'FIELD NOT NUMERIC' ACTNO
  ...
```

Output

For each record with a non-numeric field, the first line of output contains the ID field (in this case the account number) and the description. This is followed by a hexprint of the non-numeric BALANCE field.

```
10683 FIELD NOT NUMERIC

CHAR 07R6739
ZONE FFDFFFF
NUMR 0796739
     1...5..

11919 FIELD NOT NUMERIC

CHAR FFFFFFF
ZONE CCCCCCC
NUMR 6666666
     1...5..

20109 FIELD NOT NUMERIC

CHAR OO34455
ZONE DDFFFFF
NUMR 6634455
     1...5..
```

Chapter 7

Generalized/Statistical Routines O-R

This chapter lists alphabetically, and gives detailed descriptions of, routines OCCURS through REGSAMP.

OCCURS

The OCCURS routine reports on the frequency occurrence of values in a specified nonquantitative field. A report with an optional graph is produced.

Syntax

```
%OCCURS1 infile {GRAPH   {percent asterisks}}
                {NOGRAPH                    }

%OCCURS2 field
```

`infile`

Specify the name of the input file to OCCURS. A valid name is any previously defined file.

`field`

Specify the name of the nonquantitative field on which the occurrence rate is to be reported. A valid name is any nonquantitative field defined in the input file.

```
{GRAPH  }
{NOGRAPH}
```

Specify whether you want a graph produced with the report.

GRAPH—Specifies that a graph is produced. When this option is selected, the percent and asterisk parameters must be coded.

NOGRAPH—Specifies that the graph is to be omitted. When this option is selected, do not code the following two parameters.

`{percent}`

If GRAPH is specified, use this parameter to define the occurrence percentage at which graphing can begin. The percentage value is subtracted from positive graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become less than zero are set equal to zero. The percentage value is added to negative graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become greater than zero are set equal to zero.

For example, if percent is 10, graphing percentages from –10 percent to +10 percent are displayed with no asterisks. A standard graph contains no adjustment and is produced by specifying a percentage value of zero. A valid value for percentage is an actual integer value greater than or equal to zero and less than 100 or the name of a field containing an integer value in the same range. Values with decimal places are truncated on the right.

`{asterisks}`

Graph lines are drawn with the asterisk (*) character. If GRAPH is specified, the value of asterisk defines the number of asterisks that represent each percentage point. Valid values for asterisk include the actual numeric values 1 through 9 or the name of a field containing the values 1 through 9. For example, if an extremely flat distribution is anticipated, specify a higher value. This will make the graph easier to read. If a wide variance of percentages is anticipated, specify a lower value.

Operation — Stand-alone REPORT

Use the OCCURS routine to count and calculate the percentage of occurrence of a value in a specified nonquantitative field. For example, OCCURS can be used to calculate the percentage of male and female employees or the occurrence percentages of a given status code. The optional graph appears on the right side of the report, giving a graphic interpretation of the percentages. Use the INTERVL routine for analysis of quantitative fields.

Graphing

When using the percent and asterisk parameters in creating a graph, you must be careful to specify values which produce a meaningful graph. Many factors enter in to this evaluation, including the number of columns on the printer. In general:

- Where the percentage of values to be graphed is large, give the asterisks parameter a low value (1 or 2).
- If a wide variance of percentages exists, specify a low value for asterisks.

- For graphing a narrow range of percentages, increase the asterisk value (3 through 9) for easier interpretation of results.
- If the graph overflows the print line, the letter O is printed between the PCT column and the graph, indicating the overflow condition.
- The percentages calculated in the graph are the percentage of items in each occurrence grouping.

Operation — Database

OCCURS can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can either be a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of OCCURS.

Input

```
FILE PAYFILE FB (44 4400)
STATUS-CODE 1 1 N
%OCCURS1 PAYFILE GRAPH 0 3
%OCCURS2 STATUS-CODE
```

Output

This example demonstrates the calculation of the occurrence frequencies of a status code in a payroll file. Because the range of percentages is narrow, a value of 3 is selected for asterisks. While this enhances the readability of the graph, it also extends the length of each graph line. If this representation is considered too large, a value of 5 can be used for percent to effectively eliminate 15 asterisks from each graph line and shift the graph to the left.

```
                                       FREQUENCY DISTRIBUTION OF  STATUS-CODE
                                              INPUT FILENAME PAYFILE

 STATUS-CODE     COUNT     PCT

      0           55       10.7     ********************************
      1           60       11.7     ***********************************
      2           62       12.1     ************************************
      3           59       11.5     ***********************************
      4           53       10.4     *******************************
      5           57       11.1     *********************************
      6           54       10.5     ********************************
      7           49        9.6     *****************************
      8           63       12.3     *************************************
                 512      100.0
```

POPCOUNT

The POPCOUNT routine provides you with the capability of determining the population size of a file from within a user-written program or with the ATTPCT, VARPCT and the DISCPCT routines. It calculates the size of the population in the file being processed and puts the population count in a field called POP-COUNT. This field can be used in any way, but its primary purpose is as the population size parameter on ATTPCT, VARPCT and DISCPCT routines. In this way, those routines can be executed without knowing the exact population size.

Syntax

```
%POPCOUNT
```

There are no parameters for this routine.

Operation — Inline

POPCOUNT lets you determine the size of a population from within a user-written program to be used with ATTPCT, VARPCT, and DISCPCT. Each time the routine is invoked, one (1) is added to the count stored in a total field called POP-COUNT. When the file is completely processed, POP-COUNT contains the total number of records in the file. To use POPCOUNT with one of the previously mentioned CA-PanAudit Plus routines, the routines must be coded in a FINISH proc named on the JOB activity or in a subsequent JOB statement. If any records are to be eliminated from the total count, the appropriate CA-Easytrieve Plus screening logic must be inserted before POPCOUNT (see Example in this routine).

Operation — Database

No change in specification of parameters is required to use POPCOUNT with database files.

Example

The only output from POPCOUNT is the field POP-COUNT that can be used as input to the several sample routines that require population size. In this example the POP-COUNT field is used in the ATTPCT routine. Since ATTPCT is placed in a FINISH procedure, the entire file is read, and the total population size accumulated in POP-COUNT. After the entire file is processed, the FINISH procedure SAMPSIZE is executed, containing the three ATTPCT routines.

Input

```
FILE INFILE
   . . .
   ACCOUNT-STATUS   50   6   A
   . . .
*
JOB INPUT INFILE FINISH SAMPSIZE
   IF ACCOUNT-STATUS = 'CLOSED'
      GO TO JOB
   END-IF
%POPCOUNT
*
SAMPSIZE. PROC
   %ATTPCT POP-COUNT 90  1.8  3.2
   %ATTPCT POP-COUNT 95  1.8  3.2
   %ATTPCT POP-COUNT 98  1.8  3.2
END-PROC
```

Output

```
                        ATTRIBUTES SAMPLING REPORT

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT
         10,000        90           1.80           3.20     2.54000000%

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT
         10,000        95           1.80           3.20     3.54000000%

POPULATION SIZE    CONFIDENCE    PRECISION    ERROR RATE    SAMPLE PERCENT
         10,000        98           1.80           3.20     4.93000000%
```

POPSIZE

The POPSIZE routines let you determine the population size of a file from a stand-alone routine that can be used by itself or with the ATTSAMP, VARSAMP, DISCSMP, RANDPCT and RANDXCT routines. The population size is placed in a field called POP-SIZE. In this way, those sampling functions can be executed without knowing the exact population size.

Syntax

```
%POPSIZE1   infile
%POPSIZE2
```

`infile`

Specify the name of the input file to POPSIZE. A valid name is any previously defined file.

Operation — Stand-alone DISPLAY

POPSIZE lets you insert screening code between the first and second invocation of POPSIZE. This lets you bypass records that cannot be processed as part of the population.

The population size is passed by the total field called POP-SIZE. This field is then used in any of the sampling routines that require the population size to be entered.

It is important to remember that if screening code is used in POPSIZE, the same screening code must be used in the sampling routine.

Operation — Database

POPSIZE can access database and nondatabase files without any changes in the specification of parameters. The INFILE parameter can either be a nondatabase file name or the name of a database file defined in the library section.

In addition to the population size report, the field name POP-SIZE now contains the population size that can be used as input to the several sampling routines that require population size. In this example the POP-SIZE field is used to replace the population size in the ATTSAMP routine.

Input

```
FILE INFILE
...
ACCOUNTSTATUS   50   6   A
...
FILE OUTFILE
...
*
%POPSIZE1 INFILE
%POPSIZE2
*
%ATTSAMP1 INFILE POPSIZE 90 2.5 4.0 928451
%ATTSAMP2 OUTFILE
```

Output

```
POPULATION SIZE IS      10000

                   ATTRIBUTE SAMPLING REPORT

                       INPUT PARAMETERS

         INPUT FILENAME                          INFILE
         TOTAL POPULATION SIZE                   10,000
         REQUIRED PRECISION                           2.50
         REQUIRED CONFIDENCE LEVEL                   90
         ERROR RATE                                   4.00

                       SAMPLE RESULTS

         SAMPLE PERCENTAGE REQUIRED                   1.65000000%
         SAMPLE SIZE REQUIRED                       165

                       SAMPLE FILE

         NUMBER OF RECORDS PROCESSED             10,000
         NUMBER OF RECORDS REQUESTED                165
         NUMBER OF RECORDS IN SAMPLE FILE           165

         FILE OUTFILE WILL BE CREATED
```

RANDOM

The RANDOM routine uses a pseudo random number generator to produce a series of random numbers from one to 15 digits in length. The same seed will generate the same series of random numbers. Generation is initiated by a seed parameter, with different seed values resulting in different series of random numbers. The seed is used only for the first pass through the generator.

Syntax

```
%RANDOM field seed length
```

`field`

Specify the name of the field to which each number is placed after it is generated. A valid field is any previously defined field with a data type of numeric (N) or alphanumeric (A) and with length as specified in the following length parameter.

`seed`

Specify an arbitrary number, which initiates the random number generator. The seed value is related to a set of random numbers in that the same seed always produces the same set of numbers. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

`length`

Length is a numeric value from 1 to 15 that determines how many digits each random number will contain. The number specified for the length parameter must be the same as the length of the field parameter. A valid value for length is either an actual numeric value or the name of a field containing a numeric value.

Operation — Inline

RANDOM generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use RANDOM with database files.

Example

The following is an example of RANDOM.

Input

```
...
DEFINE  RANDOM-NUMBER      W  8  N
DEFINE  COUNTER            W  8  N
...
JOB INPUT NULL
DO WHILE COUNTER LT 10
   %RANDOM RANDOMNUMBER 47 8
   COUNTER  =  COUNTER  +  1
   DISPLAY +10 RANDOMNUMBER
END-DO
STOP
```

Output

This example invokes the random number generator in a DO loop to write 10 eight-digit random numbers to the field named RANDOMNUMBER. The CA-Easytrieve Plus DISPLAY statement is used to create the listing. The seed used to initiate the generation of random numbers is 47.

```
27818477
48722028
73698236
88214818
58375141
28536615
12956286
95804208
65886354
64576492
```

RANDPCT

The RANDPCT routine selects an exact percentage of records from an existing file to create a statistically valid unrestricted random sample. Selected records are optionally written to a sample file.

The sampling process begins when RANDPCT is entered for the first time. An internal table is created with one entry for each record in the file. To calculate which records are to be selected, the routine uses RANDOM to generate a record position in the file. Then, the corresponding table entry for this record is marked in the table. This is how RANDPCT keeps track of which records are to be selected.

For example, if there are 1000 records in the file, the internal table will consist of 1000 entries. If the required sample percentage is 10 percent, 100 of the 1000 entries are marked (it could be fewer than 100 if the option INC is specified for the SELECT parameter). After all selections have been made, every record with a marked table entry can be written to the sample file.

RANDPCT is used to select an exact percentage of records. To select an exact number of records, use the RANDXCT routine.

The RANDPCT routine has no limitation on the number of records in a file. However, it must dynamically obtain storage to build the internal table. The amount of storage obtained is based on the number of records in the input file. For exact details about storage requirements and other operating information, see Special Requirements in this routine.

Syntax

```
%RANDPCT1 infile size percent seed [SELECT {EXC}]
                                   [       {INC}]

%RANDPCT2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to RANDPCT. A valid name is any previously defined file.

`size`

Specify the exact number of records in the input file. For RANDPCT to select an exact percentage of records, you must specify the exact size of the file. You can enter an approximate value, but the approximation can cause a slight variance in the results. If the approximation is high, RANDPCT will expect records beyond the end of file and will select fewer than the requested percentage of records. If the approximation is low, RANDPCT selects the exact percentage of requested records but will not select records from the end of the input file. A valid value is an actual numeric value or the name of a field containing a numeric value.

The value for size determines the storage requirements for RANDPCT (Operation — Stand-alone DISPLAY).

`percent`

Indicate the percentage of records required for the sample. This value must be greater than 0 and less than 100. A valid value is an actual numeric value or the name of a field containing a numeric value.

`seed`

Specify an arbitrary number, which initiates the random number generator. The seed is used to randomize the selection of records for the sample file. Valid values include an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
[SELECT{EXC}]
[      {INC}]
```

Optionally, specify whether replacement records are selected when the random number generator selects the same record position in the internal table. It is improper for RANDPCT to write the same record to the sample file, so two procedures can be followed: either select or do not select a replacement record for the duplicate entry in the table.

EXC—Specifies exclusive selection of duplicate table entries. When a duplicate table entry is detected, another record is selected to replace it. This maintains the specified percent without writing duplicates to the sample file. EXC is the default value.

INC—Specifies inclusive selection for duplicate table entries. When a duplicate table entry is detected, another record is **not** selected to replace it. Note that this allows for the selection of less than the percentage of records specified by percent.

For example, if records 1, 3, 7, 3, and 9 are selected, RANDPCT will write records 1, 3, 7, and 9 to the output file when INC (inclusive) is specified. If EXC (exclusive) is specified, RANDPCT will select 1, 3, 7, 9, and an additional record to replace the duplicate selection of record 3.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
[DBFILE infile]
```

Optionally, specify a database file for use with RANDPCT. INFILE identifies the name of the input file to RANDPCT. The name must be the same name that was specified for INFILE on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure which is performed by the RANDPCT routine after each record is selected or not selected for the sample file. The internal field, RANDPCT-SELECTED, is set to YES if a record is selected for the sample file and to NO if a record is not selected.

After the invocation of RANDPCT2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record was selected for the sample file.

For example, the procedure could test the RANDPCT-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that RANDPCT produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify it, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You should adjust the size parameter when screening code is inserted which causes records to be bypassed from RANDPCT processing. The value specified for size must represent the size of the population being examined for RANDPCT. For example, if you specify 50,000 as the file size and 1.0 as the sample percentage, and screening code causes 25,000 of these records to be bypassed, you will get approximately 250 records in the sample file instead of the expected 500. This is because RANDPCT was invoked only 25,000 times instead of the expected 50,000 times.

To avoid this, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent a sample file from being created. Then, note the number of records processed in the RANDPCT listing. Then rerun the job using the record count listed in the report for the size parameter while specifying an output file name in place of NOFILE. This ensures that the correct percentage of records is written to the sample file while also bypassing the unwanted records.

Special Requirements

For RANDPCT to randomly select records without including duplicates, it must build an internal table to keep track of the records it has selected. The size of this table depends on the number of records in the input file (size parameter). For each 98,000 records in the input file, 12 KB of storage is dynamically obtained by RANDPCT.

DOS users must take into account the following. Depending on the value specified at installation, you may have to specify the EXITSTR parameter on the PARM statement to reserve additional storage when executing RANDPCT. This is due to the storage requirements of the internal table.

The EXITSTR parameter, of the PARM statement, specifies the storage available at execution time for user-called programs (see the CA-Easytrieve Plus *Reference Guide).*

Generally, the value for EXITSTR must be 24 KB plus the additional requirements for the internal table. However, this does not include any additional requirements that user-coded exits may require. In this case, the value should be the size calculated for the internal table plus the amount specified for EXITSTR at installation.

Operation — Database

The DBFILE parameter identifies RANDPCT as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using RANDPCT in a database application. Furthermore, you must specify all parameters on the second invocation statement, in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of RANDPCT.

Example

The following is an example of RANDPCT.

Input

```
FILE INFILE  FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE FB (44 4400)
%RANDPCT1 INFILE 10000 1.0 9383
%RANDPCT2 OUTFILE
```

Output

The report shows that a 1% sample is requested from a population of 10,000 records. The report also shows that all 10,000 records are processed and that 100 are written to the sample file. The default value of EXC was used so that duplicate table entries are replaced with new random selections. The outfile indicates that a sample file was created.

```
                    RANDPCT SAMPLING REPORT

                        INPUT PARAMETERS

          INPUT FILENAME                       INFILE
          TOTAL POPULATION SIZE                10,000
          REQUESTED SAMPLE PERCENT                  1.0000

                        SAMPLE FILE

          NUMBER OF RECORDS PROCESSED            10,000
          NUMBER OF RECORDS REQUESTED               100
          NUMBER OF RECORDS IN SAMPLE FILE          100

          FILE OUTFILE WILL BE CREATED
```

Multiple RANDPCT — Testing Subpopulations

You may want to select different percentages of samples from different subgroups of the input file. A technique called multiple RANDPCT allows you to divide the input file into subpopulations from which different percentages are randomly selected. A special method and internal fields are defined to achieve this result.

The multiple RANDPCT method is controlled by using the IF statement. IF is used to define when a new set of parameters is to be used in the sampling process. This IF statement tests the value of a user-defined counter to establish the size of a subpopulation for the random sample (see Example on the following page).

When the beginning of a new subsample is detected, specially named fields for the size and percent parameters are assigned, and a specially named field to reset the internal table must be set. The specially named fields required for the multiple RANDPCT routine are:

- RANDPCT-POPULATION– Denotes the size parameter for a particular subpopulation.
- RANDPCT-PERCENT–Denotes the required sample percentage for a particular subpopulation.
- RANDPCT-RESTART–Denotes the flag to reset the internal table.

Example

The following example demonstrates a multiple RANDPCT:

```
FILE INFILE ...
  Field-name ...
  ...
FILE OUTFILE ...
  Field-name ...
  ...
%RANDPCT1 INFILE 1000 0 17353
DEFINE COUNTER  W  4  P  0  VALUE (0)
COUNTER = COUNTER + 1
IF COUNTER EQ 1
   RANDPCT-RESTART = 'YES'
   RANDPCT-POPULATION = 200
   RANDPCT-PERCENT = 10
END-IF
IF COUNTER EQ 201
   RANDPCT-RESTART = 'YES'
   RANDPCT-POPULATION = 500
   RANDPCT-PERCENT = 5
END-IF
IF COUNTER EQ 701
   RANDPCT-RESTART = 'YES'
   RANDPCT-POPULATION = 300
   RANDPCT-PERCENT = 15
END-IF
%RANDPCT2 OUTFILE
```

RANDPCT1

The invocation of RANDPCT1 contains:

- The name of the input file
- The number of records in the file
- The value for the seed parameter
- A zero value for the percent parameter

The percent parameter is set to zero because its value will be established in subsequent coding. Next, the counter used to define the size of each individual subpopulation is defined, initialized, and incremented.

IF Statements

Each of the next three topics of code includes an IF statement, statements to assign values for the restart flag, subpopulation size and percent parameters, and an END-IF statement. You can code as many of these topics as desired. Each group of statements defines a subpopulation that will be sampled separately and written to the output file defined in RANDPCT2.

Specially Named Fields

Setting the RANDPCT-RESTART field to the value YES tells the routine to reset the internal table so that a new subsample can be selected separately from the previous sample. Any records that are selected from previous subsamples still belong to the specified sample file, but by resetting the table, the input records are treated as part of a new sample.

The RANDPCT-POPULATION field defines the size of the particular subpopulation, not the total number of records in the input file.

The RANDPCT-PERCENT field defines the required subpopulation sample percentage.

Logic

In the example, the first 200 records of INFILE are randomly sampled at a 10 percent rate, the next 500 records at a 5 percent rate, and the final 300 records at a 15 percent rate. This results in 90 records being written to the file OUTFILE, with each subsample being taken at different percentage rates.

RANDPCT2

The final statement is the invocation of RANDPCT2, which names the output file.

Coding Guidelines

The following are guidelines to be used in coding the statements that define the subpopulations:

- The condition tested in the IF statement must be true only once. If it is true more than once, the internal table is reset each time RANDPCT-RESTART is equal to the value YES, and unpredictable results will occur.
- You must ensure that the counter values tested in the IF statements are appropriate for the values specified for RANDPCT-SIZE. If the values are inappropriate, input file records may be bypassed from the selection, or the internal table may be reset before the previous subsample is completed.
- The sum of all RANDPCT-SIZE values can equal the total number of records in the input file. If not, the sample may not be a representative random sample.

RANDSPAN

The RANDSPAN routine uses a pseudo random number generator to produce a series of random numbers in a specified range. Generation is initiated by a seed parameter, with different seed values resulting in different series of random numbers. The same seed generates the same series of random numbers. The seed is used only for the first pass through the generator.

Syntax

```
%RANDSPAN    field   seed   minimum   maximum
```

`field`

Specify the name of the field to hold the generated number. A valid field is any previously defined numeric (N) or alphanumeric (A) field. The length of the field must be equal to the length of the number defined for the maximum parameter.

`seed`

Specify an arbitrary number that initiates the random number generator. The same seed always produces the same set of numbers. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

`minimum maximum`

Specify a range of numbers to be generated by the random number generator. Valid values for minimum and maximum are actual numeric values or the name of a field containing the value. Values must be no more than 15 digits in length.

Values generated are greater than or equal to the minimum or less than or equal to the maximum. For example, if the desired range is 1 through 10, specify a minimum of 1 and a maximum of 10.

Operation — Inline

RANDSPAN generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in specification of parameters is required to use RANDSPAN with database files.

Example

This example invokes the random number generator in a DO loop to write 10 random numbers in the range of 1 through 1,000 to the field named RANDOMNUMBER. The CA-Easytrieve Plus DISPLAY statement is used to create the listing. The seed used to initiate the random number generator is 999.

Input

```
...
DEFINE RANDOM-NUMBER   W     4     N
DEFINE COUNTER         W     2     N
...
JOB INPUT NULL
DO WHILE COUNTER LT 10
  %RANDSPAN RANDOM-NUMBER 999 1 1000
  COUNTER = COUNTER + 1
  DISPLAY RANDOMNUMBER
END-DO
STOP
```

Output

```
0058
0666
0647
0355
0767
0220
0002
0890
0936
0814
```

RANDXCT

The RANDXCT routine selects an exact number of records from an existing file to create a statistically valid unrestricted random sample. Selected records are optionally written to a sample file.

The sampling process begins when RANDXCT is entered for the first time. An internal table is created with one entry for each record in the file. To calculate which records are to be selected, the routine uses RANDOM to generate a record position in the file. The corresponding table entry for this record is then marked in the table. This is how RANDXCT keeps track of which records are to be selected.

For example, if there are 1000 records in the file, the internal table consists of 1000 entries. If the required sample size is 100, 100 of the 1000 entries are marked (it can be fewer than 100 if INC is specified for the SELECT parameter). After all selections are made, every record with a marked table entry can be written to the sample file.

RANDXCT is used to select an exact number of records. Use the RANDPCT routine to select an exact percentage of records.

The RANDXCT routine has no limitation on the number of records in a file. However, it must dynamically obtain storage to build the internal table. The amount of storage obtained is based on the number of records in the input file. For exact details about storage requirements and other operating information, see Special Requirements in this routine.

Syntax

```
%RANDXCT1 infile size sampsize seed [ SELECT {EXC}]
                                    [        {INC}]

%RANDXCT2 {outfile}[DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to RANDXCT. A valid name is any previously defined file.

`size`

Specify the total number of records in the input file. For RANDXCT to select an exact number of records, you must specify the exact size of the file. You can enter an approximate value, but the approximation may cause a slight variance in the results.

If the approximation is high, RANDXCT will expect records beyond the end of file and will select fewer than the requested number of records. If the approximation is low, RANDXCT selects the exact number of requested records but will not select records from the end of the input file. A valid value is an actual numeric value or the name of a field containing a numeric value.

The value for size determines storage requirements for RANDXCT. For additional information, see Operation — Stand-alone DISPLAY in this routine.

`sampsize`

Specify the number of records required for the sample. This value must be less than the size parameter. A valid value is an actual numeric value or the name of a field containing a numeric value.

`seed`

Specify an arbitrary number that initiates the random number generator. The seed is used to randomize the selection of records for the sample file. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
[SELECT {EXC}]
[       {INC}]
```

This optional parameter specifies whether replacement records are selected when the random number generator selects the same record position in the internal table. It is improper for RANDXCT to write the same record to the sample file, so two procedures can be followed: either select or do not select a replacement record for the duplicate entry in the table.

EXC—Specifies exclusive selection for duplicate table entries. When a duplicate table entry is detected, another record is selected to replace it. This maintains the exact count specified without writing duplicates to the sample file. EXC is the default value.

INC—Specifies inclusive selection for duplicate table entries. When a duplicate table entry is detected, another record is not selected to replace it.

Note: This allows for the selection of fewer than the number of records specified by sampsize.

For example, if records 1, 3, 7, 3, and 9 are selected, RANDXCT writes records 1, 3, 7, and 9 to the output file when inclusive (INC) is specified. If exclusive (EXC) is specified, RANDXCT selects 1, 3, 7, 9, and an additional record to replace the duplicate selection of record 3.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with RANDXCT. Infile identifies the name of the input file to RANDXCT. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure which is performed by the RANDXCT routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field, RANDXCT-SELECTED is set to the value YES. If a record is not selected for the sample file, RANDXCT-SELECTED is set to the value NO.

After the invocation of RANDXCT2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the RANDXCT-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that RANDXCT produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You can adjust the size parameter when screening code is inserted that causes records to be bypassed from RANDXCT processing. The value specified for the size parameter must represent the size of the population being examined for RANDXCT.

For example, if you specify 50,000 as the file size and 1,000 as the sample size and screening code causes 25,000 of these records to be bypassed, you will get approximately 500 records in the sample file instead of the expected 1,000. This is because RANDXCT is invoked only 25,000 times instead of the expected 50,000 times.

To avoid this, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent a sample file from being created. Notice the number of records processed in the RANDXCT listing. You can rerun the job using the record count listed in the report for the size parameter while specifying an output file name in place of NOFILE. This ensures that the correct number of records are written to the sample file while bypassing unwanted records.

Special Requirements

For RANDXCT to randomly select records without including duplicates, it builds an internal table to keep track of the records it has selected. The size of this table depends on the number of records in the input file (size parameter). For each 98,000 records in the input file, 12 KB of storage is dynamically obtained by RANDXCT.

DOS users must take into account the following. Depending on the value specified at installation, you may have to specify the EXITSTR parameter on the CA-Easytrieve Plus PARM statement to reserve storage when executing RANDXCT. This is due to the storage requirements of the internal table.

The EXITSTR parameter, of the PARM statement, specifies the additional storage available at execution time for user-called programs (see the CA-Easytrieve Plus *Reference Guide).*

Generally, the value for EXITSTR must be 24 KB plus the additional requirements for the internal table. However, this does not include any additional requirements that user-coded exits may require. In this case, the value can be the size calculated for the internal table plus the amount specified for EXITSTR at installation.

Operation — Database

The DBFILE parameter identifies RANDXCT as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using RANDXCT in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of RANDXCT.

Example One

The following is an example of RANDXCT.

Input

```
FILE INFILE  FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE FB (44 4400)
%RANDXCT1 INFILE 10000 120 9383
%RANDXCT2 OUTFILE
```

Output

The report shows that a sample size of 120 is requested from a population of 10,000 records. The report also shows that 10,000 records are processed and that 120 are written to the sample file. The default value of EXC is used, so duplicate table entries are replaced with new random selections. The outfile indicates that a sample file is created.

```
                    RANDXCT SAMPLING REPORT

                        INPUT PARAMETERS

          INPUT FILENAME                          INFILE
          TOTAL POPULATION SIZE                   10,000
          REQUESTED SAMPLE SIZE                      120

                        SAMPLE FILE

          NUMBER OF RECORDS PROCESSED             10,000
          NUMBER OF RECORDS REQUESTED                120
          NUMBER OF RECORDS IN SAMPLE FILE           120

          FILE OUTFILE  WILL BE CREATED
```

Multiple RANDXCT — Testing Subpopulations

You may want to select different sample sizes from different subgroups of the input file. A technique called multiple RANDXCT allows you to divide the input file into subpopulations from which different samples are randomly selected. A special method and internal fields are defined to achieve this result.

The multiple RANDXCT method is controlled by using the IF statement. IF is used to define when a new set of parameters is to be used in the sampling process. This IF statement tests the value of a user-defined counter to establish the size of a subpopulation for the random sample (see the Example on the following page).

When the beginning of a new subsample is detected, specially named fields for the size and sampsize parameters are assigned, and a specially named field to reset the internal table must be set. The specially named fields required for the multiple RANDXCT routine are:

- RANDXCT-POPULATION–Denotes the size parameter for a particular subpopulation.
- RANDXCT-SAMPSIZE–Denotes the required sample size for a particular subpopulation.
- RANDXCT-RESTART–Denotes the flag to reset the internal table.

Example

The following example demonstrates a multiple RANDXCT:

```
FILE INFILE ...
  Field-name ...
  ...
FILE OUTFILE ...
  Field-name ...
  ...
%RANDXCT1 INFILE 1000 1 17353
DEFINE COUNTER  W  4  P  0  VALUE (0)
COUNTER = COUNTER + 1
IF COUNTER EQ 1
   RANDXCT-RESTART = 'YES'
   RANDXCT-POPULATION = 200
   RANDXCT-SAMPSIZE = 20
END-IF
IF COUNTER EQ 201
   RANDXCT-RESTART = 'YES'
   RANDXCT-POPULATION = 500
   RANDXCT-SAMPSIZE = 25
END-IF
IF COUNTER EQ 701
   RANDXCT-RESTART = 'YES'
   RANDXCT-POPULATION = 300
   RANDXCT-SAMPSIZE = 45
END-IF
%RANDXCT2 OUTFILE
```

RANDXCT1

The invocation of RANDXCT1 contains:

- The name of the input file
- The number of records in the file
- A value of at least one for the sampsize parameter
- The value for the seed parameter

The value of RANDXCT-POPULATION must be at least one. The counter used to define the size of each individual subpopulation is defined, initialized, and incremented.

IF Statements

Each of the next three topics of code includes an IF statement, statements to assign values for the restart flag, subpopulation size and sampsize parameters, and an END-IF statement. You can code as many of these topics as desired. Each group of statements defines a subpopulation that will be sampled separately and written to the output file defined in RANDXCT2.

Specially Named Fields

Setting the RANDXCT-RESTART field to the value YES tells the routine to reset the internal table so that a new subsample can be selected separately from the previous sample. Any records selected from previous subsamples still belong to the specified sample file, but by resetting the table, subsequent input records are treated as part of a new sample.

The RANDXCT-POPULATION field defines the size of the particular subpopulation, not the total number of records in the input file.

The RANDXCT-SAMPSIZE field defines the required subpopulation sample size.

Logic

In the example, 20 of the first 200 records are randomly selected from INFILE, 25 of the next 500 records are selected, and 45 of the final 300 records are selected. This results in 90 records being written to the file OUTFILE, with each subsample being taken at different rates.

RANDXCT2

The final statement is the invocation of RANDXCT2, which names the output file.

Coding Guidelines

The following are guidelines to be used in coding the statements that define the subpopulations:

- The condition tested in the IF statement must be true only once. If it is true more than once, the internal table is reset each time RANDXCT-RESTART is set equal to the value YES, and unpredictable results will occur.
- You must ensure that the counter values tested in the IF statements are appropriate for the values specified for RANDXCT-SIZE. If the values are inappropriate, input file records may be bypassed from the selection or the internal table may be reset before the previous subsample is completed.
- The sum of all RANDXCT-POPULATION values can equal the total number of records in the input file. If not, the sample may not be a representative random sample.

REGEVAL

The REGEVAL routine is used with the REGSAM and REGSAMP routines. It is the final step in regression estimation and evaluates the sample selected by the REGSAMP routine. REGEVAL provides a report showing the estimate of the total audited amount and the achieved precision, based on the REGSAMP sample.

Syntax

```
%REGEVAL1 infile field1 field2 size conf prec total
```

```
%REGEVAL2
```

`infile`

This parameter names the input file to REGEVAL, which is the file produced by the REGSAMP routine. A valid name is any previously defined file.

`field1`

Specify the name of the quantitative field containing the recorded amount for each record in the REGSAMP file. A valid name is any quantitative field defined in the input file.

`field2`

Specify the name of the quantitative field containing the audited amount for each record in the REGSAMP file. A valid name is any quantitative field defined in the input file.

`size`

Specify the number of records in the original file, not the number of records in the REGSAMP file. A valid value is an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify the confidence level. Confidence is a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ from the result that can be obtained by examining the entire population by more than the specified precision.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. This parameter can be specified as an actual numeric value or the name of a field containing a valid numeric value.

The value must be the same as the value specified for confidence in the REGSAMP routine.

`precision`

Specify a value for precision. Precision is a quantitative tolerance range, such as an implied plus (+) and minus (-) amount. The difference between the sample results and results that can be obtained from examining the entire file can fall in this range at the specified confidence level. A valid value is either an actual numeric value or the name of a field containing a numeric value with up to two decimal places. Values greater than two decimal places are truncated on the right.

The value must be the same as the value specified for precision in the REGSAMP routine.

`total`

Specify the total recorded amount of all records in the original file. A valid value is either an actual numeric value or the name of a field containing a numeric value.

Operation — Stand-alone DISPLAY

If the results of using REGEVAL are to be sufficiently valid, the REGSAMP sample file must contain at least 20 differences in the recorded and audited amounts. If there are fewer than 20 differences, an approximation method is used.

Operation — Database

REGEVAL cannot be used in a database application.

Example

The following is an example of REGEVAL, the final step in regression estimation.

Input

```
FILE SAMPLE ...
  RECAMT ...
  AUDAMT ...
  ...
%REGEVAL1 SAMPLE RECAMT AUDAMT 1000 99 506885 506885591.00
%REGEVAL2
```

Output

The report shows the input parameters. The report also shows the estimated standard deviation and standard error of the regression estimate of the REGSAMP sample. The final result shows the achieved precision and estimated audited amount.

From the output, you can see that the estimated audited amount is in the achieved precision of the total population recorded amount.

```
                          REGRESSION ESTIMATION EVALUATION

                                  INPUT PARAMETERS

INPUT FILENAME                                                      SAMPLE
RECORDED AMOUNT FIELD                                               RECAMT
AUDITED AMOUNT FIELD                                                AUDAMT
TOTAL POPULATION SIZE                                                1,000
CONFIDENCE LEVEL                                                       99%
DESIRED PRECISION                                               506,885.00
TOTAL POPULATION RECORDED AMOUNT                             50,688,591.00
```

```
                                 EVALUATION RESULTS

SAMPLE SIZE                                                             51
ESTIMATED STANDARD DEVIATION OF REGRESSION                          420.12
STANDARD ERROR OF REGRESSION ESTIMATE                            58,828.85
REGRESSION COEFFICIENT                                                1.00
ACHIEVED PRECISION                                              151,777.70
ESTIMATED AUDITED AMOUNT                                     50,747,756.94
```

REGSAM

The REGSAM routine is used with the REGSAMP and REGEVAL routines. REGSAM determines the estimated standard deviation of the original population.

Syntax

```
%REGSAM1 infile field1 field2 size
```

```
%REGSAM2
```

`infile`

Specify the name of the input file to REGSAM. This must be the name of the preliminary sample file created for estimating the standard deviation of the original population. A valid name is a previously defined file.

`field1`

Specify the name of the quantitative field containing the recorded amount for each record in the preliminary sample file. A valid name is any quantitative field defined in the input file.

`field2`

This parameter names the quantitative field containing the audited amount for each record in the preliminary sample file. A valid name is any quantitative field defined in the input file.

`size`

Specify the total number of records in the original population file. This is not the number of records in the preliminary sample file. A valid value is either an actual numeric value or the name of a field containing a numeric value.

Operation — Stand-alone DISPLAY

The input file to REGSAM is the preliminary sample file described under the subject Regression Estimation in the chapter "Using Routines," earlier in this guide. Attribute sampling methods can be used to determine an appropriate size for this preliminary sample. Notice that the file size in REGSAM is the size of the original file, not the preliminary sample file.

Operation — Database

REGSAM cannot be used in a database application.

Example

The following is an example of REGSAM, which is the first step in regression estimation.

Input

```
FILE PRELIM ...
  RECAMT ...
  AUDAMT ...
  ...
%REGSAM1 PRELIM RECAMT AUDAMT 1000
%REGSAM2
```

Output

The estimated standard deviation in the output is used as a parameter for REGSAMP, which is the next step in the regression estimation. The standard error of the regression is a value that states the range of error of the file total that can be possible based on the estimated standard deviation.

```
                    UNSTRATIFIED VARIABLE SAMPLING
                     REGRESSION ESTIMATION METHOD

                       INPUT PARAMETERS

INPUT FILENAME                                           PRELIM
RECORDED AMOUNT FIELD                                    RECAMT
AUDITED AMOUNT FIELD                                     AUDAMT
TOTAL POPULATION SIZE                                     1,000

                      ESTIMATION RESULTS

ESTIMATED STANDARD DEVIATION OF REGRESSION                1,437.60
STANDARD ERROR OF THE REGRESSION                        143,760.40
```

REGSAMP

The REGSAMP routine is used with the REGSAM and REGEVAL routines. REGSAMP calculates the percentage of the original file's total records that constitutes a representative sample. It then randomly selects the appropriate number of records from the file.

The calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Estimated standard deviation

Selected records can be written to a sample file. A report lists the input parameters and the result of the sample size calculation.

Syntax

```
%REGSAMP1 infile size confidence precision stddev seed

%REGSAMP2 {outfile}  [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to REGSAMP, which is the original file. A valid name is any previously defined file.

`size`

Specify the number of records in the original file. A valid value is either an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify the confidence level. Confidence is a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample and the result obtained by examining the entire population does not differ by more than the specified precision.

For example, a confidence level of 90 means that there are 90 chances in 100 that the sample is representative and 10 chances that it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. This parameter can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`precision`

Specify the precision. Precision is a quantitative tolerance range, such as an implied plus (+) and minus (-) amount. The difference between the sample results and results that can be obtained from examining the entire file can fall in this range at the specified confidence level.

A valid value is an actual numeric value or the name of a field containing a numeric value. It can contain up to two decimal places. Values greater than two decimal places are truncated on the right.

`stddev`

Specify the estimated standard deviation amount calculated by using the REGSAM routine (see REGSAM routine). A valid value for stddev is either an actual numeric value or the name of a field containing a numeric value. This value is limited to nine digits, which includes two decimal places. Values greater than nine digits are truncated on the left. Values greater than two decimal places are truncated on the right.

`seed`

Specify an arbitrary number that initiates the random number generator. This seed is used to randomize the selection of samples from the file. A valid value is either an actual numeric value or the name of a field containing a numeric value. Values can be the name of a field containing a numeric value up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with REGSAMP. Infile identifies the name of the input file to REGSAMP. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure that is performed by the REGSAMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field REGSAMP-SELECTED is set to the value YES. If a record is not selected for the sample file, REGSAMP-SELECTED is set to the value NO.

After the invocation of REGSAMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the REGSAMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that REGSAMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You can adjust the size parameter when screening code is inserted that causes records to be bypassed from REGSAMP processing. The value specified for size must represent the size of the population being examined for the regression estimation. For the resulting sample percentage and optional sample file to be accurate, the size parameter must reflect any records that are bypassed.

For example, if you specify 50,000 as the file size, and logic prior to REGSAMP causes 25,000 records to be bypassed, the population size sampled is actually only 25,000. The calculated percentage is therefore incorrect, and any sample file created is also invalid.

To avoid this, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent the sample file from being created. Notice the number of records processed by REGSAMP listed in the report. You can rerun the job using the record count provided in the report for the size parameter, while specifying an output file name in place of NOFILE. This ensures the correct results from REGSAMP processing, while bypassing the unwanted records.

Notice that the value for precision is specified as an absolute amount, not as a percentage.

Operation — Database

The DBFILE parameter identifies REGSAMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using REGSAMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of REGSAMP.

Example

The following is an example of REGSAMP.

Input

```
FILE ORIGINL ...
  fieldname ...
  ...
FILE SAMPLE ...
  fieldname ...
  ...
%REGSAMP1 ORIGINL 1000 99 504178 2013.76 9
%REGSAMP2 SAMPLE
```

Output

The report shows the input parameters, the results of the REGSAMP calculations, and the results of the sampling process, including whether an output file was created.

This file is then used as input to the REGEVAL routine, which is the third step in regression estimation.

```
                UNSTRATIFIED VARIABLE SAMPLING
                 REGRESSION ESTIMATION METHOD

                    INPUT PARAMETERS

INPUT FILENAME                       ORIGINL
TOTAL POPULATION SIZE                  1,000
REQUIRED CONFIDENCE LEVEL                 99%
REQUIRED PRECISION                   506,885.00
STANDARD DEVIATION OF REGRESSION       1,437.60

                    SAMPLE RESULTS

SAMPLE PERCENTAGE REQUIRED                 5.0821%
SAMPLE SIZE REQUIRED                      51

                     SAMPLE FILE

NUMBER OF RECORDS PROCESSED            1,000
NUMBER OF RECORDS REQUESTED               51
NUMBER OF RECORDS IN SAMPLE FILE          51

FILE SAMPLE   WILL BE CREATED
```

Chapter 8

Generalized/Statistical Routines S-Z

This chapter lists alphabetically, and gives detailed descriptions of, routines SIMPREG through WEEKDAY.

SIMPREG

The SIMPREG routine performs a simple linear regression and correlation analysis. This routine solves the regression equation:

```
y = a + bx
```

Syntax

```
%SIMPREG1 infile field1 field2
%SIMPREG2
```

`infile`

Specify the name of the input file to SIMPREG. A valid name is any previously defined file.

`field1`

Specify the name of the field containing the y value in the previous equation. A valid value is any numeric field defined in the input file. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`field2`

Specify the name of the field containing the x value in the previous equation. A valid value is any numeric field defined in the input file. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

Operation — Stand-alone DISPLAY

The variables to be analyzed must be contained on the input file with each record containing the corresponding x and y values.

Operation — Database

SIMPREG can access database and nondatabase files without any changes in the specification of parameters. The infile parameter can be either a nondatabase file name or the name of a database file defined in the library section.

Example

The following is an example of SIMPREG.

Input

```
FILE INFILE FB (20 2000)
XVALUE 1 8 N 2
YVALUE 9 8 N 2
%SIMPREG1 INFILE YVALUE XVALUE
%SIMPREG2
```

Output

```
                            SIMPLE REGRESSION ANALYSIS

                                 INPUT PARAMETERS

                       INPUT FILENAME                         INFILE
                       Y FIELD                                YVALUE
                       X FIELD                                XVALUE

REGRESSION EQUATION
A VALUE (Y INTERCEPT)              2.91
B VALUE (SLOPE)                    1.28

                               MEAN                   STD DEV
X VALUE                       149.01                  29.14
Y VALUE                       193.45                  45.82

CORRELATION ANALYSIS
MULTIPLE R                        .8132
R SQUARE                          .6613
STD ERROR OF EST                   26.73

NUMBER OF RECORDS             200
```

The SIMPREG report shows the coefficients a and b that solve the previous equation for the x and y values input. The value for a is the y intercept of the regression line, which is the value of y when x is equal to 0. The value for b is known as the slope of the regression line and represents the amount of change in y when x is increased by one unit.

The report also presents an analysis of the two variables x and y, including their mean, standard deviation, and the regression of y on x.

The statistics presented in the correlation analysis demonstrate the relationship of the variables to the regression line. The multiple R value and the corresponding R SQUARE are an expression of the correlation between the regression line and the actual x and y values. Values for R range between 1.0 and +1.0, with 1.0 being maximum inverse correlation and +1.0 maximum positive correlation.

R SQUARE is another measurement of correlation, called the coefficient of determination, which ranges between 0 and +1.0. It is a more meaningful indicator of the relationship between y and x than is the R value.

The standard error of estimate is the possible error of the regression analysis on all input values in the file.

SPS

The SPS (sampling proportional to size) routine creates a sample file where the probability of selecting a record for the sample is proportional to the size of the value in a specified field. The probability that a record will be selected for sampling depends on whether the accumulated value of a specified numeric field is the first to exceed a target value.

The method of operation is as follows. The random number generator is used to initialize a work field, which is incremented by the absolute, actual, or positive value in the specified field. Input values are accumulated until the total exceeds a target value. The record of which value caused the target amount to be exceeded is written to the sample file. The accumulator is then reset to a value between zero and the target value, and the process continues. See Operation — Stand-alone DISPLAY in this routine.

Syntax

```
%SPS1 infile field target seed  [VALUE {ABS}]
                                [      {ACT}]
                                [      {POS}]

%SPS2 {outfile{ [DBFILE infile] [PERFORM procname]
      {NOFILE }
```

`infile`

Specify the name of the input file to SPS. A valid name is any previously defined file.

`field`

Specify the name of the field of which value is accumulated to reach the target value. A valid name is any numeric field defined in the input file.

`target`

Specify a value which, when exceeded, causes the record to be written to the sample file. A valid value is an actual numeric value or the name of a field containing a numeric value.

`seed`

Specify an arbitrary number that initiates the random number generator. This seed is used to initialize a work field, which accumulates the input value. A valid value is an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
[VALUE{ABS}]
[     {ACT}]
[     {POS}]
```

This optional parameter controls the value for the input field used in the sampling process. The default value is ABS.

ABS—Specifies that the absolute value of the input field is used in the sampling process. This method places equal emphasis on both positive and negative values.

ACT—Specifies that the actual value of the input field is used in the sampling process. This method places emphasis on the net value of positive and negative values.

POS—Specifies that only the values of the input field that are greater than zero are used in the sampling process. This method places emphasis on the positive values.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with SPS. Infile identifies the name of the input file to SPS. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure that is performed by the SPS routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field SPS-SELECTED is set to the value YES. If a record is not selected for the sample file, SPS-SELECTED is set to the value NO.

After the invocation of SPS2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the SPS-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that SPS produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

The value of the input field is added to the accumulator. When the accumulator exceeds the target value, it is reduced to a value between zero and the target value. Depending on the size of the value that caused the target to be exceeded, the accumulator can exceed the target by a small or a large amount. The accumulator is reduced by enough multiples of the target value to ensure that the new value of the accumulator is greater than 0 and less than the target value.

Exceeding the target by varying amounts greater than the target has no effect on future sampling. The accumulator is reset to a value less than the target amount every time. For example, if the target is 10000, and the accumulated value is 11134, the accumulator is reset to 1134 (reduced by 10000). If the accumulator is 32532, it is reset to 2532 (reduced by 30000).

Operation — Database

The DBFILE parameter identifies SPS as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using SPS in a database application. Furthermore, specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of SPS.

Example

The following is an example of SPS.

Input

```
FILE INFILE ...
  BALANCE ...
  ...
FILE OUTFILE ...
  Field-name ...
  ...
%SPS1 INFILE BALANCE 10000 97
%SPS2 OUTFILE
```

Output

This example demonstrates the selection of a sample from a customer file of current account balances. The total of these balances is $779,451.62. The accumulation of the input values caused the target value of 10,000 to be exceeded 76 times, which results in 76 records being written to the output file.

```
                   SPS SAMPLING REPORT

                     INPUT PARAMETERS

INPUT FILENAME                                    INFILE
INPUT FIELD                                      BALANCE
VALUE OR INPUT FIELD IS                              ABS
TARGET VALUE                                      10,000.00

                     SAMPLE FILE

NUMBER OF RECORDS PROCESSED                        1,342
ABS VALUE OF RECORDS PROCESSED                   647,786.79
ACT VALUE OF RECORDS PROCESSED                   632,296.81
POS VALUE OF RECORDS PROCESSED                   640,041.80
NUMBER OF RECORDS IN SAMPLE FILE                      64

FILE OUTFILE WILL BE CREATED
```

SQRT

The SQRT routine calculates the square root of a specified number. The result is accurate to two decimal places.

Syntax

```
%SQRT number result
```

`number`

Specify the numeric value on which the square root calculation is to be performed. A valid value for the number parameter is either an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`result`

Specify the name of the field to which the result of the calculation is placed. A valid name is any previously defined numeric field.

Operation — Inline

SQRT generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use SQRT with database files.

Example

The following is an example of SQRT. This example demonstrates the use of SQRT to calculate the radius for a circle of a given area. The formula for radius given the area is:

```
RADIUS = SQRT(AREA/PI).
```

Input

```
FILE ...
  AREA          1   8  P  2
  RADIUS        W   6  P  2
JOB ...
DEFINE WORKAREA      W   8   P   2
DEFINE PI            W   3   P   4   VALUE (3.1416)
WORKAREA  =  AREA  /  PI
%SQRT WORKAREA RADIUS
PRINT RADIUS-REPORT
...
REPORT RADIUS-REPORT LINESIZE 72
TITLE 1 'CALCULATION OF RADIUS'
LINE AREA RADIUS
```

Output

The radius is calculated for each area in the input file, and the area and radius are printed in the RADIUS-REPORT. The field named WORKAREA is used to hold the value of (AREA/PI) for the square root calculation.

```
           CALCULATION OF RADIUS

            AREA            RADIUS

      2,345,154.00          863.99
         35,355.78          106.08
          1,968.92           25.03
             .            .
             .            .
             .            .
            400.00           11.28
```

SRCECOMP

The SRCECOMP routine compares two versions of a source program to determine the differences between them. It prints a listing of the programs or a report of all added, deleted, moved, and changed statements.

Syntax

```
%SRCECOMP oldfile oldfield newfile newfield {ALL    }
                                            {CHANGES}
```

`oldfile`

Specify the name of the file containing the old source statements to be compared. A valid name is any previously defined file.

`oldfield`

Specify the name of a field defined in the file identified by oldfile. The location and attributes of this field determine where the comparison will begin for the source statements in the old file and the length of the comparison.

A valid name for oldfield is any field defined in oldfile. It must have a length attribute of less than 255 and the same length attribute as newfield. The starting location of the field can differ from that of newfield.

`newfile`

Specify the name of the file containing the new source statements to be compared. A valid name is any previously defined file.

`newfield`

Specify the name of a field defined within the file indicated by newfile. The location and attributes of this field determine where the comparison will begin for the source statements in the new file and the length of the comparison.

A valid name for newfield is any field defined in newfile. It must have a length attribute of less than 255 and the same length attribute as oldfield. The starting location of the field can differ from that of oldfield.

```
{ALL    }
{CHANGES}
```

This parameter specifies what will be printed in the SRCECOMP report.

ALL—Indicates that both changed and unchanged statements will be listed.

CHANGES—Indicates that only changed statements will be listed.

Operation — Stand-alone REPORT

The SRCECOMP routine can compare a maximum combination of 32,000 identical, added, or deleted statements in the two files. It assigns sequence numbers to statements in the programs being compared. On the report, these numbers appear under the columns OLD SEQ NUM (for statements that appear in the old version) and NEW SEQ NUM (for statements that appear in the new version). These represent the statement's relative position in each file. The maximum length of a record in oldfile or newfile is 6136.

The report lists a maximum of 105 source characters. If the source statements contain more than 105 characters, the listing prints only the first 105 characters of the statement.

Screening of input data in SRCECOMP is not allowed. If screening is required, code the logic to screen an input file in another job step and write the desired records to a temporary file. Then use this temporary file as input to SRCECOMP.

SRCECOMP requires 400 KB of storage for execution.

Operation — Database

SRCECOMP cannot be used in a database application.

Example

The following is an example of SRCECOMP.

Input

```
FILE OLDFILE ...
  COMPARE-OLD ...
FILE NEWFILE ...
  COMPARE-NEW ...
...
%SRCECOMP OLDFILE COMPARE - OLD NEWFILE COMPARE - NEW CHANGES
```

Output

This report was generated by the CHANGES option of SRCECOMP and lists only the statements that have been changed. It lists the statement number; whether it has been added, deleted, or moved; where it has been moved from or to; and the first 105 bytes of the statement.

```
    4/20/88          SOURCE COMPARE         PAGE      1


   OLD    NEW
   SEQ    SEQ
   NUM    NUM
 SOURCE

    2         DELETED        PAYDEPT  29  2  N
            2 ADDED          SORTFIELD  29  2  N
            7 MVD FROM     8 ZONE  27 2 N
    8         MVD TO       7 ZONE  27 2 N
           10 MVD FROM    12 DEFINE INTAMT W 6 P 2
   12         MVD TO      10 DEFINE INTAMT W 6 P 2
   15         DELETED        DEFINE FIRSTSW  W  1 N VALUE (0)
           15 ADDED          DEFINE SWITCH   W  1 N VALUE (0)
   17         DELETED        SKIP 1
   18         DELETED        SORT PAYFILE TO VFMFILE USING (PAYDEPT)
           17 ADDED          SORT PAYFILE TO VFMFILE USING (SORTFIELD)
   24         DELETED           IF FIRSTSW EQ 1
           23 ADDED             IF SWITCH EQ 1
   27         DELETED           FIRSTSW = 1
           32 ADDED             SWITCH = 1
   35         DELETED           OLDDEPT = DEPT
   36         MVD TO      39    INDICATOR = 2
           34 ADDED             INTAMT = INTAMT + PAY
           35 MVD FROM    40    INDICATOR = 1
   40         MVD TO      35    INDICATOR = 1
           39 MVD FROM    36    INDICATOR = 2
```

STDDEV

The STDDEV routine is used to calculate the standard deviation of a set of numbers.

Standard deviation is a statistical value that gives a measurement of the variability of a set of numbers. The greater the variability, the larger the standard deviation.

In most applications for standard deviation, the file is divided into groups or intervals based on the value of a control field. The standard deviation of these intervals and the standard deviation of the entire population are used for statistical purposes and in other CA-PanAudit Plus routines.

To calculate the standard deviation, you:

1. Obtain the difference between the value of each item in the population and the population mean.
2. Square the difference.
3. Add them together.
4. Divide the sum by the total number of items.
5. Extract the square root.

Standard deviation is a required input to the CA-PanAudit Plus statistical routines VARSAMP and VARPCT and can be obtained for a variety of other statistical purposes.

Syntax

```
%STDDEV field indicator intervaldev totaldev
```

`field`

Specify the name of the quantitative field for which the standard deviation is calculated. A valid field name is any previously defined quantitative field. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`indicator`

The indicator parameter is used to inform STDDEV when an input value represents the start of a new interval or when it is just another item in the interval. The method used is as follows.

For indicator, you specify a one-character work field, which is used to indicate three different calculation options to STDDEV (see Option 1, Option 2, and Option 3). These options work with the following two parameters to regulate the initialization of internal work fields.

Any job that uses STDDEV must contain logic to control the process of calculating the standard deviation of intervals in a file and/or the standard deviation of the entire population. Intervals must be sequenced; see Operation — Inline on the following page.

In some situations, you may not want the population standard deviation to reflect the values from all intervals. In this case, use the indicator parameter to reset to zero the internal work fields for both the interval and total standard deviation.

The following values for indicator control the calculations for the intervaldev and totaldev parameters.

Option 1- Initializing the Work Fields (0)

Set the indicator field to zero when you want to reset both the totaldev and intervaldev fields to zero.

Option 2 - Calculating Intervals (1)

Set the indicator field to one at the beginning of each new interval. This initializes the intervaldev field to zero and begins the calculation for the new interval mean.

Option 3 - Calculating Intervals and Whole Populations (2)

Set the indicator field to two for the second and all subsequent records in that interval. This causes the STDDEV routine to update the values for intervaldev and totaldev.

`intervaldev`

Specify the name of a user-defined, quantitative, W-type work field with two decimal places. The standard deviation for the interval is maintained in this field. (Working storage is discussed in the CA-Easytrieve Plus *Reference Guide.)*

When the first record for an interval is detected (when the indicator is set to one), the STDDEV routine initializes intervaldev to zero. When the second and all subsequent records for an interval are processed (when the indicator is two), intervaldev is updated to reflect the standard deviation of the current field parameter value plus the values from all previously processed records in that interval.

`totaldev`

Specify the name of a user-defined, quantitative, W-type work field with two decimal places. Standard deviation for the entire population is maintained in this field.

Each time STDDEV processes a record, totaldev is updated to reflect the standard deviation of the current field parameter value plus the values of all previous records, unless the indicator parameter has been set to zero. If the indicator parameter has been set to zero, totaldev reflects the standard deviation of all values occurring after the indicator is set to a nonzero value.

Operation — Inline

If you want interval standard deviation processing, records input to STDDEV must be in sequence according to what comprises the intervals. For example, if each department comprises an interval, records must be sorted according to department. If each interval is a range of dollar amounts, records must be sorted according to those dollar amounts. Sequencing in CA-Easytrieve Plus is performed by the SORT statement. For additional information, see the CA-Easytrieve Plus *Reference Guide.*

Operation — Database

No change in the specification of parameters is required to use STDDEV with database files.

Examples

The following two examples are invocations of STDDEV which demonstrate different uses of the indicator parameter.

Example One

In this example, the standard deviation is determined for intervals of the field PAY based on the control field DEPT.

The following provides a brief description of the important fields defined in this example:

INTAMT—The interval total of the field for which the standard deviation is being calculated

TOTAMT—The field total for the whole file

INTDEV—The standard deviation of each interval

TOTDEV—The standard deviation for the entire file

OLDDEPT—A working storage field that contains the control value of the last interval

FIRSTSW—A working storage field that contains the value zero for the first record processed and the value one for all subsequent records

INDICATOR—The indicator parameter

Input

```
FILE PAYFILE  FB(20 2000)
      SORTFIELD      7  3  N
    FILE VFMFILE FB(20 2000) VIRTUAL
      PAY      1   6   P   2
      DEPT     7   3   N
      DEFINE INTAMT     W  8  P  2. DEFINE TOTAMT    W  8  P  2
      DEFINE INTDEV     W  8  P  2. DEFINE TOTDEV    W  8  P  2
      DEFINE OLDDEPT    W  3  N  VALUE (0)
      DEFINE FIRSTSW    W  1  N  VALUE (0)
      DEFINE INDICATOR  S  1  N
    SORT PAYFILE TO VFMFILE USING (SORTFIELD)
    JOB INPUT VFMFILE   FINISH END-OF-JOB
      IF PAY NOT NUMERIC
         GO TO JOB
      END-IF
      IF DEPT NE OLDDEPT
         IF FIRSTSW EQ 1
            PRINT STDDEV-REPORT
         END-IF
         PERFORM NEW-INTERVAL
      ELSE
         PERFORM OLD-INTERVAL
      END-IF
      PERFORM STANDARD-DEVIATION
      NEW-INTERVAL. PROC
         FIRSTSW  =  1
         INTAMT  =  PAY
         OLDDEPT  =  DEPT
         INDICATOR  =  1
      END-PROC
      OLD-INTERVAL. PROC
          INTAMT  =  INTAMT  +  PAY
          INDICATOR  =  2
      END-PROC
      STANDARD-DEVIATION. PROC
         TOTAMT  =  TOTAMT  +  PAY
         %STDDEV PAY INDICATOR INTDEV TOTDEV
      END-PROC
      END-OF-JOB. PROC
         PRINT STDDEV-REPORT
      END-PROC
    REPORT STDDEV-REPORT
      TITLE 1 'STANDARD DEVIATION OF PAYFILE BY DEPARTMENT'
      HEADING OLDDEPT ('DEPT.')
      HEADING INTAMT ('TOTAL', 'FOR DEPARTMENT')
      HEADING INTDEV ('STD. DEVIATION', 'BY DEPARTMENT')
      HEADING TOTAMT ('TOTAL', 'FOR COMPANY')
      HEADING TOTDEV ('STD. DEVIATION', 'FOR COMPANY')
      LINE OLDDEPT INTAMT INTDEV TOTAMT TOTDEV
```

Output

```
                STANDARD DEVIATION OF PAYFILE BY DEPARTMENT

           TOTAL         STD. DEVIATION        TOTAL         STD. DEVIATION
DEPT. FOR DEPARTMENT    BY DEPARTMENT      FOR COMPANY       FOR COMPANY

 01      6,978,481.91       54,559.37       6,978,481.91       54,559.37
 02      7,468,652.90       58,502.32      14,447,134.81       56,667.54
 05      7,981,761.60       58,211.22      22,428,896.41       57,401.78
 07      7,240,895.64       55,674.23      29,669,792.05       56,984.89
 12      6,531,546.62       56,750.47      36,201,338.67       57,119.18
 14      7,222,257.51       56,266.94      43,423,596.18       56,979.45
 21      7,057,234.07       58,402.64      50,480,830.25       57,187.24
```

In this example, you begin by defining the library section, and continue by defining all necessary working storage fields. The file PAYFILE is sorted to the CA-Easytrieve Plus virtual file VFMFILE. (See the CA-Easytrieve Plus *Reference Guide* for a discussion of virtual files.) The sorted VFMFILE then becomes the input file to the job that computes the standard deviation. The logic is as follows:

- If the PAY field contains a non-numeric value, the GO TO JOB statement bypasses this particular record.
- The input record is read, and the DEPT field is compared to OLDDEPT, which is initialized to zero (it is assumed that no DEPT value of zero exists). If the values are not equal, a new department has been found, and the FIRSTSW field is compared to the value one.

  If this is not the first record (FIRSTSW = 1), a line of the report is printed because you have just encountered a new interval (department number). The NEW-INTERVAL procedure is then performed, which sets INDICATOR to one. This tells STDDEV that a new interval has been encountered.
- If a new department is not found, the procedure, OLD-INTERVAL, is performed. This sets INDICATOR to two, which tells STDDEV that the value being processed is in the current interval.
- The STANDARD-DEVIATION procedure is performed, which invokes STDDEV with the correct value for INDICATOR. When the last record has been read, the procedure END-OF-JOB, listed on the JOB statement, prints data for the last interval of the report.

The output lists the total and standard deviation for each department. The running total for the file and the accumulated standard deviation are also listed when each department line is printed. The file total and total standard deviation are the last numbers in the last two columns.

Example Two

In this example, the standard deviation of the PAY field for the total file is determined. Interval values are not calculated.

The following is a brief description of the important fields defined in this example:

INTDEV–A dummy field for the invocation of STDDEV

TOTAMT–The field total for the whole file

TOTDEV–The standard deviation for the entire file

INDICATOR–The indicator parameter

FIRSTSW–A working storage field that contains the value zero for the first record processed and the value one for all subsequent records

Input

```
FILE ...
  PAY      1   6   P   2
  DEFINE INTAMT     W  8  P  2. DEFINE TOTAMT    W  8  P  2
  DEFINE INTDEV     W  8  P  2. DEFINE TOTDEV    W  8  P  2
  DEFINE INDICATOR  S  1  N.   DEFINE FIRSTSW   W  1  N  VALUE (0)
JOB INPUT PAYFILE   FINISH END-OF-JOB
  IF PAY NOT NUMERIC
     GO TO JOB
  END-IF
  IF FIRSTSW EQ 0
     INDICATOR  =  1
     FIRSTSW  =  1
  ELSE
     INDICATOR  =  2
  END-IF
  PERFORM STANDARD-DEVIATION
  STANDARD-DEVIATION. PROC
     TOTAMT  =  TOTAMT  +  PAY
     %STDDEV PAY INDICATOR INTDEV TOTDEV
  END-PROC
  END-OF-JOB. PROC
     PRINT STDDEV-REPORT
  END-PROC
REPORT STDDEV-REPORT
  TITLE 1 'STANDARD DEVIATION OF PAYFILE'
  HEADING TOTAMT ('TOTAL', 'FOR COMPANY')
  HEADING TOTDEV ('STD. DEVIATION', 'FOR COMPANY')
  LINE TOTAMT TOTDEV
```

Output

```
STANDARD DEVIATION OF PAYFILE

        TOTAL          STD. DEVIATION
     FOR COMPANY         FOR COMPANY

     49,760,795.68         57,936.50
```

As shown, you begin by defining the library section and continue with the definition of all necessary working storage fields.

The INTDEV field is not used in the execution of the job, but it still must be defined as a working storage field and listed as a parameter on the invocation for STDDEV. Because there is no interval processing, you need not sort the input file. The logic is as follows:

- If the PAY field contains a non-numeric value, the GO TO JOB statement bypasses this particular record.
- If the FIRSTSW field indicates that this is the first record, INDICATOR is set to one. For all other records, INDICATOR is set to two.
- The STANDARD-DEVIATION procedure is performed, which invokes the STDDEV routine.
- When the last record has been read, the procedure END-OF-JOB, listed on the JOB statement, prints the STDDEV-REPORT.

Because this example is run against the same file as Example One, the totals printed in this report are identical to the final totals in the last two columns from Example One.

STOPORGO

The STOPORGO routine performs a Stop or Go Sampling that selects multiple statistically valid random samples from a single input file. Each individual sample is independently written to a specified output file. You can specify up to five output files.

The sampling process begins when STOPORGO is entered for the first time. An internal table is created with one entry for each record in the file. To calculate which records are to be selected, the routine uses RANDOM to generate a record position in the file. The corresponding table entry for this record is then marked in the table. This is how STOPORGO keeps track of which records are to be selected for a particular output file.

For example, if there are 1000 records in the file, the internal table consists of 1000 entries. If a required sample size is 100, 100 of the 1000 entries are marked (it can be fewer than 100 if INC is specified for the SELECT parameter). After all selections have been made for every output file, every record with a marked table entry is written to the appropriate sample file.

The STOPORGO routine has no limitation on the number of records in a file. However, to accomplish this, it must dynamically obtain storage to build the internal table. The amount of storage obtained is based on the number of records in the input file. For exact details about storage requirements and other operating information, see Special Requirements in this routine.

Syntax

```
%STOPORGO infile size seed numsamp [SELECT {EXC}] SS1 samp1     +
                                   [       {INC}]

[SS2 samp2] [SS3 samp3] [SS4 samp4] [SS5 samp5] OUT1 outfile1   +

[OUT2 outfile2] [OUT3 outfile3] [OUT4 outfile4] [OUT5 outfile5] +

[PERFORM procname]
```

`infile`

Specify the name of the input file to STOPORGO. A valid name is any previously defined file.

`size`

Specify the total number of records in the input file. For STOPORGO to select an exact number of records, you must specify the exact size of the file. You can enter an approximate value, but the approximation may cause a slight variance in the results. If the approximation is high, STOPORGO expects records beyond the end-of-file and will select fewer than the requested number of records.

If the approximation is low, STOPORGO selects the exact number of requested records but does not select records from the end of the input file. A valid value is either an actual numeric value or the name of a field containing a numeric value.

The value for size determines storage requirements for STOPORGO. For additional information, see Special Requirements in this routine.

`seed`

Specify an arbitrary number that initiates the random number generator. This seed is used to randomize the selection of records for the sample files. A valid value is either an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

`numsamp`

Specify the number of output files you want to be produced. The value for numsamp must equal the number of keyword parameters SSn and OUTn. Numsamp can be an actual numeric value or the name of a field containing a numeric value. Valid values are the numbers 1 through 5.

```
[SELECT{EXC}]
[      {INC}]
```

This optional parameter specifies the way that replacement records are selected when the random number generator selects the same record position in the internal table. It is incorrect for STOPORGO to write the same record to a sample file; so two procedures can be followed: either select or do not select a replacement record for the duplicate entry in the table.

EXC—Specifies exclusive selection of duplicate table entries. When a duplicate table entry is detected, another record is selected to replace it. This maintains the exact count specified without writing duplicates to the sample file. EXC is the default value.

INC—Specifies inclusive selection for duplicate table entries. When a duplicate table entry is detected, another record is **not** selected to replace it. This allows for the selection of fewer than the number of records specified by sampn.

For example, if records numbered 1, 3, 7, 3, and 9 are selected, STOPORGO will write records 1, 3, 7, and 9 to the output file when inclusive (INC) is specified. If exclusive (EXC) is specified, STOPORGO will select 1, 3, 7, 9, and an additional record to replace the duplicate selection of record 3.

`SS1 samp1 [SSn sampn]`

Specify the number of records required for each sample. Specify a value for sampn that corresponds to the desired sample size for outfilen. A valid value is an actual numeric value or the name of a field containing a numeric value. You can specify sample sizes for up to five output files.

Note: You must specify a value for samp1. Specification of values for samp2 through samp5 is optional.

`OUT1 outfile 1 [OUTn outfilen]`

Specify the name of the output file to which records for each sample are to be written. The name specified for outfilen corresponds to the sample size specified for SSn sampn. File characteristics must be coded on the FILE statement for this output file. Outfilen must have the same file characteristics as the input file, or outfilen must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfilen include any previously defined file. You can specify up to five output files.

Note: You must specify a name for outfile1. Specification of file names for OUT2 through OUT5 is optional. The number of keyword parameters SSn and OUTn must be equal to the value given for numsamp.

`[PERFORM procname]`

Specify the name of a CA-Easytrieve Plus procedure that is performed by the STOPORGO routine after each record is selected or not selected for an output file. If a record is selected for an output file, an internal field is set to the value YES. If a record is not selected for an output file, an internal field is set to the value NO. Internal field names are defined for each output file and are named STOPORGOn-SELECTED, where n is the number of the output file as defined in the OUTn parameter.

After the invocation of STOPORGO, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for an output file.

For example, the procedure can test the STOPORGO1-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all records selected for the output file specified in the OUT1 parameter in addition to the normal report that STOPORGO produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

Screening of input data in STOPORGO is not allowed. If screening is required, code the logic to screen the input records in a previous job step and write the desired records to a temporary file. Then use this temporary file as input to STOPORGO.

Operation — Database

STOPORGO cannot be used in a database application without extracting data to a sequential file.

Special Requirements

For STOPORGO to randomly select records without including duplicates, it builds an internal table to keep track of the records it has selected. The size of this table depends on the number of records in the input file (the size parameter). For each 98,000 records in the input file, 12 KB of storage is dynamically obtained by STOPORGO.

VSE users must take into account the following consideration. Depending on the value specified at installation, you may have to specify the EXITSTR parameter on the CA-Easytrieve Plus PARM statement to reserve storage space when executing STOPORGO. This is due to the storage requirements of the internal table.

The EXITSTR parameter of the PARM statement specifies the additional storage available at execution time for user-called programs. For additional information, see the CA-Easytrieve Plus *Reference Guide.*

Generally, the value for EXITSTR must be 24 KB plus the additional requirements for the internal table. However, this does not include any additional requirements that user-coded exits may require. In this case, the value can be the size calculated for the internal table plus the amount specified for EXITSTR at installation.

Example

The following is an example of STOPORGO.

Input

```
PARM ABEXIT SNAP
FILE INFILE  FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFIL1 FB (44 4400)
FILE OUTFIL2 FB (44 4400)
FILE OUTFIL3 FB (44 4400)
FILE OUTFIL4 FB (44 4400)
%STOPORGO INFILE 15000 19387 4 SS1 100 SS2 200 SS3 250 SS4 500   -
          OUT1 OUTFIL1 OUT2 OUTFIL2 OUT3 OUTFIL3 OUT4 OUTFIL4
```

Output

```
                    STOPORGO SAMPLING REPORT

                        INPUT PARAMETERS

INPUT FILENAME                              INFILE
TOTAL POPULATION SIZE                       15,000
NUMBER OF SAMPLE FILES                           4

                        SAMPLE FILE(S)

                    OUTPUT FILE NUMBER  1

NUMBER OF RECORDS PROCESSED                 15,000
NUMBER OF RECORDS REQUESTED                    100
NUMBER OF RECORDS IN SAMPLE FILE               100

FILE OUTFIL1 WILL BE CREATED

                    OUTPUT FILE NUMBER  2

NUMBER OF RECORDS PROCESSED                 15,000
NUMBER OF RECORDS REQUESTED                    200
NUMBER OF RECORDS IN SAMPLE FILE               200

FILE OUTFIL2 WILL BE CREATED

                    OUTPUT FILE NUMBER  3

NUMBER OF RECORDS PROCESSED                 15,000
NUMBER OF RECORDS REQUESTED                    250
NUMBER OF RECORDS IN SAMPLE FILE               250

FILE OUTFIL3 WILL BE CREATED

                    OUTPUT FILE NUMBER  4

NUMBER OF RECORDS PROCESSED                 15,000
NUMBER OF RECORDS REQUESTED                    500
NUMBER OF RECORDS IN SAMPLE FILE               500

FILE OUTFIL4 WILL BE CREATED
```

This example shows that four samples were requested from the input file named infile. It then shows that 15,000 records were processed for each sample and that four sample files containing 100, 200, 250, and 500 records are produced.

STRATIF

The STRATIF routine provides a report that describes a recommended sample file utilizing a stratified random sampling method, and optionally creates the sample file. The report details the following:

- Range of values in each stratum
- Number of records in each stratum
- Total amount of items in each stratum
- Standard deviation of values in each stratum
- Sample size for each stratum
- Percentage of the stratum requested by the sample

Syntax

```
[%STRATTAB number1 ... number255]

%STRATIF1 infile field {target  } conf prec {materiality}   +
                       {STRATTAB}           {STRATTAB   }
          seed [MAXSTRATA value]  [LRECL length]

%STRATIF2 {outfile} {STRTEVL}  [DBFILE infile]
          {NOFILE } {NOFILE }
          {       } {       }
```

`[number1 ... number255]`

Specify actual numeric values that will be used as multiple-defined end points. Required when STRATTAB is specified as the target value. There is a maximum of 255 end points.

`infile`

Specify the name of the input file to STRATIF. A valid name is any previously defined file.

`field`

Specify the name of the quantitative field from which the values for the stratification are taken. A valid name is any quantitative field defined in the input file.

```
{target  }
{STRATTAB}
```

Target is used to separate the input file into individual strata and represents the approximate total value of all values in the stratum. The exact use of this parameter is discussed in the topic Target Value in the subject Stratified Random Sampling. Valid values can contain up to two decimal places and are truncated on the right if more than two are specified.

Specify the keyword STRATTAB to indicate multiple end points. The optional routine %STRATTAB must be coded if STRATTAB is specified.

`conf`

Specify the confidence level. Confidence is a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ by more than the specified precision from results that can be obtained by examining the entire population.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances that it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. Confidence can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`prec`

Specify the precision. Precision is a quantitative tolerance range, such as an implied plus (+) and minus (-) amount. The difference between the sample results and results that can be obtained from examining the entire file can fall in this range at the specified confidence level. Precision can be an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

```
{materiality}
{STRATTAB   }
```

Specify a value for materiality. Materiality is the value that determines whether items will be stratified for sampling or selected for a separate stratum of all items greater than materiality. If an input value is less than or equal to materiality, it participates in the stratification process. If it is greater than materiality, it is selected for what is, in effect, a 100 percent sample. Valid values can contain up to two decimal places. Values greater than two decimal places are truncated on the right.

Specify STRATTAB if the highest end point value is to be used for materiality.

`seed`

Specify an arbitrary number that initiates the random number generator. The seed is used for the RANDXCT routine, which randomly selects the samples for each stratum. Seed can be an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

`[MAXSTRATA value]`

MAXSTRATA defines maximum number of strata. The default maximum number of strata permitted is 256. The value of MAXSTRATA must be a positive, nonzero integer.

The maximum number of strata includes the following:

- The negative stratum
- The zero stratum
- All strata between the zero and materiality strata
- Materiality stratum

If the number of strata created exceeds the value of MAXSTRATA, message PAP311 is issued, and STRATIF terminates. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*. For information regarding the number of strata and its implications in stratified random sampling, see the chapter "Using Routines."

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + 81 work bytes + 4 RDW bytes = LRECL
```

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are to be written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

```
{STRTEVL}
{NOFILE }
```

STRTEVL—Specify this optional parameter only when the STRTEVL routine is to be used to perform an evaluation of the sample file. For STRTEVL to perform its calculations, you must save the internal table generated by STRATIF. When STRTEVL is specified, STRATIF writes the internal table to a file with a DD (DLBL) name of STRTBL.

See OS/390, z/OS, and VSE examples of the user coded JCL for this file in this routine. If this parameter is not specified, the table is not written to the STRTBL file and no STRTEVL is possible.

NOFILE—This option is valid only for database use of STRATIF. NOFILE specifies that the internal table for STRTEVL will not be written to the STRTBL file, and no evaluation is possible.

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with STRATIF. Infile identifies the name of the input file to STRATIF. The name must be the same name that you specified for infile on the first invocation statement.

Operation — Stand-alone REPORT

Selecting the Target Value

The specification of a proper target value is important to the creation of a meaningful report and sample file from STRATIF. One method of selecting a target value is:

- To aid in selecting a target value, use the INTERVL routine to study the distribution of the input file.
- After you select a target value, run STRATIF using the NOFILE parameter to study the report without creating a sample file.
- Adjust the target value to accommodate the improved stratification plan or to eliminate a partially filled next-to-last stratum. See the discussion of Target Value in the subject Stratified Random Sampling in the chapter "Using Routines."

It is not unusual to run STRATIF two or three times to fine tune the target value. When you achieve a satisfactory stratification, you can substitute the outfile parameter for NOFILE and create the sample file.

Using the STRATIF Routine

Many types of statistical analysis are available for calculating the number of samples for a given stratum. STRATIF uses the mean estimation method, which uses the stratum population size and the stratum standard deviation in calculating the sample size. The RANDXCT routine is then used to select the appropriate number of samples from each stratum population by random sampling.

For the STRTEVL evaluation routine to be performed on the sample file, you must code a user-defined file in the JCL. This file must be a sequential disk file with the DD (DLBL) name of STRTBL and will contain one 128-byte record for each stratum generated by STRATIF. The following contains OS/390 and z/OS as well as VSE examples of JCL for the required file.

Example OS/390 and z/OS JCL

```
//STEP1    EXEC PGM=EZTPA00
//STRTBL   DD DSN=user.file.name,DISP=(NEW,CATALOG),UNIT=SYSDA,
//            SPACE=(TRK,(5,1))
              .
              .
  OTHER NECESSARY JCL
              .
              .
//SYSIN    DD *
FILE INFILE ...
  Field-name ...
  ...
FILE OUTFILE ...
  Field-name ...
  ...
%STRATIF1 INFILE BALANCE 100000 95 10000 50000 1357531
%STRATIF2 OUTFILE STRTEVL
```

Example VSE JCL

```
// JOB STRATIF
// ASSGN SYSnnn,DISK,VOL=xxxxxx,SHR
// DLBL STRTBL,'user.file.name',99/365,SD
// EXTENT SYSnnn,xxxxxx,1,0,start,length
       .
       .
   OTHER NECESSARY JCL
       .
       .
// EXEC EZTPA00
FILE INFILE ...
  Field-name ...
  ...
FILE OUTFILE ...
  Field-name ...
  ...
%STRATIF1 INFILE BALANCE 100000 95 10000 50000 1357531
%STRATIF2 OUTFILE STRTEVL
/*
/&
```

Operation — Database

The DBFILE parameter identifies STRATIF as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, when you use STRATIF in a database application, specify all parameters on the second invocation statement, in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of STRATIF.

Examples

The following are two examples of STRATIF.

Example One

This example demonstrates the use of the STRATIF routine.

Input

```
FILE CUSTFIL FB (44 4400)
 BALANCE 1 8  P  2
 FILE SAMPFIL FB (44 4400)
 %STRATIF1 CUSTFIL BALANCE 2700000 95 1000000 50000 1357531
 %STRATIF2 SAMPFIL
```

Output

```
6/28/90                          RESULTS OF STRATIFIED SAMPLING                PAGE     1
                 INPUT FILENAME: SAMPLE    INPUT FIELD: BALANCE
                 STRATUM SIZE:          2,700,000.00    MATERIALITY:            50,000.00
                                   PRECISION: 1,000,000.00    CONFIDENCE:95%

                                                                     STD        SAMP     PCT
FROM                TO              FREQ           TOTAL             DEV        SIZE     INT

        1.00-            .01-         0              .00              .00         0        .0
         .00             .00           0              .00              .00         0        .0
         .01          700.79       6,736     2,700,376.86           172.88        17        .2
      700.80          993.99       3,186     2,700,241.07            85.02         4        .1
      994.00       14,022.29         425     2,706,994.81         4,083.16        26       6.1
   14,022.30       25,636.14         157     2,708,264.22         2,225.47         5       3.1
   25,636.15       40,976.07          79     2,711,335.44         4,609.24         5       6.3
   40,976.08       50,000.00          54     2,478,950.88         2,548.02         2       3.7
   50,000.01       99,979.11         283    21,187,908.17        15,056.46       283     100.0

FINAL TOTAL                       10,920    37,194,071.45        12,867.63       342       3.1

                                  THE RECOMMENDED SAMPLE SIZE LESS MATERIALITY         59

                 NO OUTPUT FILE OF SELECTED RECORDS WILL BE PRODUCED
```

The first two strata listed in the output are for values less than zero and equal to zero. The input file contained no values with these characteristics, so no stratum statistics exist for these strata. If values in these strata did exist, they cannot contribute to the stratified sampling calculations. All zero and less than zero values are shown in the report, but they are ignored for sampling purposes.

The other strata contain enough input values to create a total stratum value of almost exactly 2,700,000 (target value), except for the last two strata. The last stratum is the top stratum and contains all values above 50,000 (materiality), and its total is in excess of the target value. The next to last stratum contains only 2,478,950.88 because no more records exist on the input file.

The original target value chosen for this example was 2,500,000. This created a next-to-last stratum that was only 34 percent full and resulted in a sample size of zero for this stratum. This is undesirable because all stratum should contribute to the sample file with at least one, and preferably two items. The target value was fine tuned to 2,700,000 to achieve a 92 percent full next-to-last stratum and resulted in the selection of three items for the sample file. This fine tuning ensures accurate results for both the sample file and for the STRTEVL routine.

Recommended sample size less materiality represents the size of the sample from all strata other than the top stratum. The PCT INT column represents the percentage of items in the stratum that are selected for the sample file.

Example Two

This example demonstrates the use of STRATIF with the STRATTAB parameter.

Input

```
FILE CUSTFIL  FB (44 4400)
BALANCE 1  8  P  2
FILE SAMPFIL  FB (44 4400)
%STRATTAB 244 292 314 366 387 461 628 736 760 1000
%STRATIF1 CUSTFIL BALANCE STRATTAB 99 50 STRATTAB 23465
%STRATIF2 SAMPFIL
```

Output

```
                                RESULTS OF STRATIFIED SAMPLING                     PAGE      1
                                INPUT FILENAME: CUSTFIL    INPUT FIELD: BALANCE
                                              VARIABLE STRATUM SIZE
                                    PRECISION:       50.00     CONFIDENCE:99%

                                                                        STD        SAMP      PCT
FROM                    TO                FREQ            TOTAL         DEV        SIZE      INT

        1.00-                  .01-         0               .00          .00          0        .0
         .00                   .00            0               .00          .00          0        .0
         .01                244.00        1,567        269,676.71        41.75      1,235      78.8
      244.01                292.00          492        131,979.88        13.76        128      26.0
      292.01                314.00          245         74,279.63         5.90         27      11.0
      314.01                366.00          568        193,542.04        15.08        162      28.5
      366.01                387.00          222         83,603.71         5.73         24      10.8
      387.01                461.00          766        325,555.41        21.51        311      40.6
      461.01                628.00        1,863      1,014,984.90        49.20      1,730      92.8
      628.01                736.00        1,201        818,817.74        30.80        698      58.1
      736.01                760.00          284        212,459.58         6.81         36      12.6
      760.00              1,000.00        2,792      2,457,568.99        69.61      2,792     100.0
    1,000.01  9,999,999,999,999.99          920     32,344,933.03    30,467.70        920     100.0

FINAL TOTAL                              10,920     37,927,401.62    13,062.45      8,063      73.8

                                 THE RECOMMENDED SAMPLE SIZE LESS MATERIALITY               7,143
```

STRTEVL

The STRTEVL routine evaluates the results of the stratified random sample created by the STRATIF routine. The recorded and audited amounts for the items in the sample file are input to STRTEVL to calculate an estimated audited amount of the sample and the achieved precision of that estimate.

Syntax

```
%STRTEVL1 infile field1 field2  [LRECL length]
%STRTEVL2
```

`infile`

Specify the name of the input file to STRTEVL. A valid name is any previously defined file.

`field1`

Specify the name of the quantitative field containing the recorded amount for each record. A valid name is any quantitative field defined in the input file.

`field2`

Specify the name of the quantitative field containing the audited amount for each record. A valid name is any quantitative field defined in the input file.

`[LRECL length]`

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record using the following formula:

```
Infile-lrecl + lenfield1 + lenfield2+ 1 work byte + 4 RDW bytes = LRECL
```

Operation — Stand-alone DISPLAY

To evaluate the sample file from STRATIF using STRTEVL, you must enter the audited amounts and the corresponding recorded amounts into the sample records through an edit facility such as TSO. To ensure accurate results with STRTEVL you must enter all audited amounts that correspond to all recorded amounts in the sample file.

Values for confidence, precision, and materiality are used in the calculations for STRTEVL. These are obtained from the table generated during the execution of STRATIF. The table specified in the JCL for STRTEVL must be the same table created by STRATIF and used in the generation of the sample file by the STRATIF routine. This table (STRTBL) must be specified as the DD (or DLBL) name on the JCL statement. The following contains OS/390 and z/OS as well as VSE examples of JCL for the required file.

The STRTEVL routine creates a CA-Easytrieve Plus table. Depending on the input data, the default allocation of 256 table entries may be exceeded. Error message A008 will inform you of this condition. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*. To increase the allocation for table entries, the CA-Easytrieve Plus options table must be link edited with a new maximum value. For details, see the CA-Easytrieve Plus *Getting Started* guide.

Example OS/390 and z/OS JCL

```
//STEP1    EXEC PGM=EZTPA00
//STRTBL   DD DSN=user.file.name,DISP=(OLD,KEEP)
             .
             .
  OTHER NECESSARY JCL
             .
             .
//SYSIN    DD *
FILE SAMPFIL ...
  BALANCE ...
  AUDAMT ...
  ...
%STRTEVL1 SAMPFIL BALANCE AUDAMT
%STRTEVL2
```

Example VSE JCL

```
// JOB STRTEVL
// ASSGN SYSnnn,DISK,VOL=xxxxxx,SHR
// DLBL STRTBL,'user.file.name',99/365,SD
// EXTENT SYSnnn,xxxxxx,1,0,start,length
        .
        .
   OTHER NECESSARY JCL
        .
        .
// EXEC EZTPA00
FILE SAMPFIL ...
  BALANCE ...
  AUDAMT ...
  ...
%STRTEVL1 SAMPFIL BALANCE AUDAMT
%STRTEVL2
/*
/&
```

Operation — Database

STRTEVL cannot be used in a database application.

Example

The following are input statements to STRTEVL and the resulting output. The sample file produced by the STRATIF example routine was updated with hypothetical audited amounts to obtain the results shown.

Input

```
FILE SAMPFIL
  BALANCE        1   8   P   2
  AUDAMT         9   8   P   2
%STRTEVL1 SAMPFIL BALANCE AUDAMT
%STRTEVL2
```

Output

```
                    PARAMETERS USED BY STRTEVL

                         INPUT PARAMETERS

INPUT FILENAME:                                     SAMPFIL
RECORDED AMOUNT FIELD:                              BALANCE
AUDITED AMOUNT FIELD:                                AUDAMT

                 VALUES OBTAINED FROM STRATIF

CONFIDENCE:                                              95
PRECISION:                                     1,000,000.00
MATERIALITY:                                      50,000.00

                STRATIFIED VARIABLE SAMPLING
                    EVALUATION PROCEDURE

ESTIMATED AUDITED AMOUNT                         15,888,227.64
ACHIEVED PRECISION                                1,030,246.93
TOTAL AUDITED AMOUNT MATERIALITY STRATUM         21,285,723.52
```

In this example, the estimated audited amount is the total of the file that can be projected on the total population based on the input of the audited amounts. The estimated audited amount printed by STRTEVL does not include the audited amounts for the top stratum (items greater than materiality). To calculate the estimated audited amount for the entire file, you must add the audited amount for the top stratum to the estimated audited amount printed by STRTEVL.

Achieved precision is the precision for the total file based on audited amounts from the sample file. To maintain the confidence and precision specified in the STRATIF and STRTEVL routines, the difference between the actual file total and the estimated audited amount for the entire file must be within the value of achieved precision.

The example shows that the audited amount from the top stratum is 21,285,723.52. When this is added to the estimated audited amount from STRTEVL, the total is 37,173,951.16. This is within the achieved precision (1,030,246.93) of the actual file total (37,194,071.45). (The actual file total can be found in the example output for STRATIF. The audited amount for the top stratum is obtained through separate calculation of the audited top stratum items.)

TIMECONV

The TIMECONV routine converts time represented in hundredths of a second to a representation including hours, minutes, seconds, and hundredths of seconds.

Syntax

```
%TIMECONV time1 time2
```

`time1`

Specify the name of the field to be converted. This field must contain time represented in hundredths of seconds. A valid name is any previously defined numeric field.

`time2`

Specify the name of the field to which the result of the time conversion will be placed. Results are placed in this field in the format HHMMSShh. A valid name is any previously defined numeric field that will hold eight characters.

Operation — Inline

TIMECONV generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use TIMECONV with database files.

Example

The following is an example of TIMECONV.

Input

```
FILE INFILE CARD
  TIMEIN  1  8  N 0
  TIMEOUT        W  8 N MASK 'ZZ:ZZ:ZZ.99'
  HOURS   TIMEOUT  2   N
  MINUTES TIMEOUT +2 2 N
  SECONDS TIMEOUT +4 2 N
  TENTHS  TIMEOUT +6 2 N
JOB INPUT INFILE
  %TIMECONV TIMEIN TIMEOUT
  PRINT RPT1
REPORT RPT1 LINESIZE 72
  LINE TIMEIN TIMEOUT HOURS MINUTES SECONDS TENTHS
END
00000001
00000010
00000100
00001234
00060000
00120000
00240000
10601234
/*
```

Output

The TIMEIN field is converted from hundredths of a second to hours, minutes, seconds, and hundredths of a second. After the calculation is made, the converted time is placed in the TIMEOUT field.

```
        TIMEIN        TIMEOUT     HOURS   MINUTES    SECONDS   TENTHS
             1            .01      00        00         00        01
            10            .10      00        00         00        10
           100           1.00      00        00         01        00
         1,234          12.34      00        00         12        34
        60,000       10:00.00      00        10         00        00
       120,000       20:00.00      00        20         00        00
       240,000       40:00.00      00        40         00        00
    10,601,234    29:26:52.34      29        26         52        34
```

UNBYTE

Information is stored and transmitted on a computer's magnetic devices by a two-state data representation. The presence or absence of magnetized spots on the surface of a revolving magnetic disk or reel of magnetic tape represents the value of zero (absence) or value of one (presence) to the computer. Each piece of information is referred to as a binary digit, or bit. A sequence of eight adjacent bits forms one byte.

Because a byte is the smallest addressable unit on the computer, UNBYTE, the bit manipulation routine, makes it easier for you to access encoded information on a bit-for-bit basis. UNBYTE provides a method of investigating all eight bits of a given byte.

For example, a 0 in a certain position of a byte can be used to indicate male, and a 1 can indicate female. The UNBYTE routine takes an input field and creates nine fields, named BIT0, BIT1, BIT2, . . . , BIT7, and ALLBITS. Fields BIT0 through BIT7 are one byte fields that correspond to the value (0 or 1) of bits 0 through 7 of the input byte. ALLBITS is an eight byte field that contains the values of each of the individual fields BIT0 through BIT7.

Syntax

```
%UNBYTE inputbyte
```

`inputbyte`

Specify the name of a field of which bits are placed in fields BIT0 through BIT7 and ALLBITS, as discussed previously. A valid name is any previously defined field. The field must be only one byte in length.

Operation — Inline

The UNBYTE routine can be used as often as required to interrogate as many bytes as necessary. However, each time it executes, the BIT0 through BIT7 and ALLBITS fields are reused. These fields are defined by UNBYTE and must not be defined by you.

Since the INPUTBYTE is restricted to being one byte in length, the CA-Easytrieve Plus overlay redefinition concept can be used to define a one-byte field from a multibyte field. For details, see the topic on the DEFINE Statement in the CA-Easytrieve Plus *Reference Guide*.

UNBYTE generates no output and can be used alone or with other routines and/or CA-Easytrieve Plus logic.

Operation — Database

No change in the specification of parameters is required to use UNBYTE with database files.

Example

The following is an example of UNBYTE.

Input

```
FILE PAYROLL
  PAY-STATUS      1   1   B
DEFINE MALE-COUNTER      W   3   P   0
DEFINE FEMALE-COUNTER    W   3   P   0
JOB INPUT PAYROLL FINISH END-OF-JOB
%UNBYTE PAY-STATUS
IF BIT4 EQ 0
   MALE-COUNTER = MALE-COUNTER + 1
ELSE
   FEMALE-COUNTER = FEMALE-COUNTER + 1
END-IF
END-OF-JOB. PROC
DISPLAY 'PAYROLL DEPARTMENT REPORT'
DISPLAY SKIP 1, 'NUMBER OF MALES =', MALECOUNTER
DISPLAY 'NUMBER OF FEMALES =', FEMALECOUNTER
END-PROC
```

Output

This sample examines bit 4 of the PAYSTATUS byte of a payroll file, counts the number of male and female employees, and prints the totals after all payroll records have been examined.

```
PAYROLL DEPARTMENT REPORT

NUMBER OF MALES   =   118
NUMBER OF FEMALES =    94
```

VARPCT

The VARPCT routine calculates the percentage of the total number of records in a file which constitutes a representative sample. The calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Standard deviation

A report lists the input parameters and the calculated sample percentage.

Use the VARSAMP routine to calculate the appropriate sample size and then randomly select the records from a file.

Syntax

```
%VARPCT size confidence precision stddev
```

`size`

Specify the total number of records in the population being examined. A valid value is either an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ by more than the specified precision from the result that can be obtained by examining the entire population.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances that it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. Confidence can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`precision`

Specify a quantitative tolerance range in the calculations, such as an implied plus (+) and minus (-) amount. The difference between the sample results and results that can be obtained from examining the entire file can fall in this range, at the specified confidence level. Precision can be an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two decimal places are specified.

`stddev`

Specify the standard deviation of the field on which the variables sampling is being conducted. The standard deviation is a measure of the variability of a set of numbers and can be calculated using the STDDEV routine. A valid value for stddev is an actual numeric value or the name of a field containing a numeric value. Values can contain up to nine digits, which include two decimal places. Values greater than nine digits are truncated on the left; values with more than two decimal places are truncated on the right.

Operation — Inline

VARPCT provides you with the ability to study the result of variable sampling with a particular set of parameters without reading an input file. This provides a technique for studying results so that you can make the best possible parameter selection for the given application. VARPCT can be invoked any number of times, allowing you to study the effects of varying the parameters (see Example). For example, you can use it to evaluate the effects on the sample size of varying the precision percentage.

When satisfactory results have been obtained with VARPCT, you can select the proper parameters and run the VARSAMP routine (that produces identical results) to randomly select the desired samples from a file.

If VARPCT is used with a JOB INPUT NULL statement, the job must contain a STOP statement.

Note: The value for precision is specified as an absolute amount. (Precision in the ATTPCT and ATTSAMP routines, described earlier in this chapter, is expressed as a percentage.)

Operation — Database

No change in the specification of parameters is required to use VARPCT with database files.

Example

The following is an example of VARPCT.

Input

```
JOB INPUT NULL
...
%VARPCT 20000 95 9000000 3000.00
%VARPCT 20000 95 10000000 3000.00
%VARPCT 20000 95 11000000 3000.00
...
STOP
```

Output

```
                              VARIABLES SAMPLING REPORT

  POPULATION SIZE  CONFIDENCE     PRECISION       STANDARD DEVIATION       SAMPLE
                                                                          PERCENT

           20,000      95         9,000,000.00             3,000.00        0.8450

  POPULATION SIZE  CONFIDENCE     PRECISION       STANDARD DEVIATION       SAMPLE
                                                                          PERCENT

           20,000      95        10,000,000.00             3,000.00        0.6850

  POPULATION SIZE  CONFIDENCE     PRECISION       STANDARD DEVIATION       SAMPLE
                                                                          PERCENT

           20,000      95        11,000,000.00             3,000.00        0.5650
```

This example demonstrates a technique for evaluating the effects on the sample percentage by varying the precision. The input code consists of three invocations of VARPCT with identical values for population size, confidence, and standard deviation, but with precisions of 9, 10, and 11 million. As the precision amount increases (that represents a decrease in sampling precision), the required sample percentage decreases. The report lists the input values and the different sample percents for the values specified.

VARSAMP

The VARSAMP routine calculates the percentage of a file's total records that constitutes a representative sample and then randomly selects the appropriate number of records from the file.

The calculation for sample size is based on four statistical parameters:

- File size
- Desired confidence level
- Precision
- Standard deviation

Selected records can be written to a sample file. A report lists the input parameters and the result of the sample size calculation.

Use the VARPCT routine to calculate the appropriate sample size without selecting records.

Syntax

```
%VARSAMP1 infile size confidence precision stddev seed

%VARSAMP2 {outfile} [DBFILE infile] [PERFORM procname]
          {NOFILE }
```

`infile`

Specify the name of the input file to VARSAMP. A valid name is any previously defined file.

`size`

Specify the total number of records in the population being examined. A valid value is either an actual numeric value or the name of a field containing a numeric value.

`confidence`

Specify a numeric value that represents the confidence percentage, such as the probability that the result obtained from the sample does not differ by more than the specified precision from the result that can be obtained by examining the entire population.

For example, a confidence level of 90 means there are 90 chances in 100 that the sample is representative and 10 chances that it is not representative. The confidence percentage must be one of the following: 50, 68, 75, 80, 85, 90, 95, 96, 97, 98, or 99. Confidence can be specified as an actual numeric value or the name of a field containing a valid numeric value.

`precision`

Specify a quantitative tolerance range in the calculations, such as an implied plus (+) and minus (-) amount. The difference between the sample results and results that can be obtained from examining the entire file can fall in this range, at the specified confidence level. Precision can be an actual numeric value or the name of a field containing a numeric value. Values can contain up to two decimal places and are truncated on the right if more than two are specified.

`stddev`

Specify the standard deviation of the field on which the variables sampling is being conducted. The standard deviation is a measure of the variability of a set of numbers and can be calculated using the STDDEV routine. A valid value for stddev is an actual numeric value or the name of a field containing a numeric value. Values can contain up to nine digits, which include two decimal places. Values greater than nine digits are truncated on the left; values with more than two decimal places are truncated on the right.

`seed`

Specify an arbitrary number that initiates the random number generator. The seed is used to randomize the selection of samples from the file. A valid value is either an actual numeric value or the name of a field containing a numeric value. Values can be up to seven digits in length with no decimal places. Values greater than seven digits are truncated on the left.

```
{outfile}
{NOFILE }
```

Specify whether records selected for the sample are to be written to an output file.

outfile—Records selected for the sample are written to the output file indicated by outfile. File characteristics must be coded on the FILE statement for this output file. Outfile must have the same file characteristics as the input file, or outfile must have the appropriate file characteristics to be able to accommodate the longest input record. Valid names for outfile include any previously defined file.

NOFILE—Records selected for the sample are not written to an output file.

`[DBFILE infile]`

This optional parameter specifies a database file for use with VARSAMP. Infile identifies the name of the input file to VARSAMP. The name must be the same name that you specified for infile on the first invocation statement.

```
[PERFORM procname]
```

Specify the name of a CA-Easytrieve Plus procedure that is performed by the VARSAMP routine after each record is selected or not selected for the sample file. If a record is selected for the sample file, the internal field VARSAMP-SELECTED is set to the value YES. If a record is not selected for the sample file, VARSAMP-SELECTED is set to the value NO.

After the invocation of VARSAMP2, you can define a CA-Easytrieve Plus procedure to perform processing based on whether the input record is selected for the sample file.

For example, the procedure can test the VARSAMP-SELECTED field and display appropriate fields of the input record if the value is YES. This provides a listing of all selected records in addition to the normal report that VARSAMP produces. For a description of the format and use of a procedure, see the CA-Easytrieve Plus *Reference Guide.* For an example of the use of this parameter, see the chapter "Advanced Techniques."

This is an optional parameter. If you do not specify the name, the system substitutes a default procname which is a dummy procedure that performs no processing.

Operation — Stand-alone DISPLAY

You can adjust the size parameter when screening code is inserted that causes records to be bypassed from VARSAMP processing. The value specified for the size parameter must represent the size of the population being examined for the variables sampling. For the resulting sample percentage and optional sample file to be accurate, the size parameter must be adjusted by the number of records that are bypassed.

For example, if you specify 50,000 as the file size, and screening code causes 25,000 of these records to be bypassed, the population size sampled by VARSAMP is actually 25,000. The calculated percentage is therefore incorrect, and the sample file created is also invalid.

To avoid this result, whenever screening code bypasses records, specify the actual file size for the size parameter and the NOFILE option to prevent a sample file from being created. Notice the number of records processed by VARSAMP in the report. You can rerun the job, specifying for size the record count listed in the report, and specify an output file name in place of NOFILE. This ensures the correct results from VARSAMP processing while bypassing the unwanted records.

Additional parameters may also require adjustment if records are bypassed, such as the precision or stddev parameters.

Notice that the value for precision is specified as an absolute amount. (Precision in the ATTPCT and ATTSAMP routines, described earlier in this chapter is expressed as a percentage.)

Operation — Database

The DBFILE parameter identifies VARSAMP as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when using VARSAMP in a database application. Furthermore, you must specify all parameters on the second invocation statement in the order shown in the description of the syntax. This restriction on parameter placement applies only to the database use of VARSAMP.

Example

The following is an example of VARSAMP.

Input

```
FILE INFILE FB (44 4400)
NAME 1 15 A
BIRTH 16 6 N MASK('Z9/99/99')
EMPLOYED 22 5 N
ZONE  27 2 N
DEPT 29 2 N
GROSS 31 14 N 2
FILE OUTFILE FB (44 4400)
%VARSAMP1 INFILE 2000 98 100000000 259469.41 1357
%VARSAMP2 OUTFILE
```

Output

```
                    VARIABLES SAMPLING REPORT

                        INPUT PARAMETERS

INPUT FILENAME                              INFILE
TOTAL POPULATION SIZE                        2,000
REQUIRED CONFIDENCE LEVEL                       98
REQUIRED PRECISION                      100,000,000.00
STANDARD DEVIATION                          259,469.41

                        SAMPLE RESULTS

SAMPLE PERCENTAGE REQUIRED                      6.8000%
SAMPLE SIZE REQUIRED                          136

                        SAMPLE FILE

NUMBER OF RECORDS PROCESSED                  2,000
NUMBER OF RECORDS REQUESTED                    136
NUMBER OF RECORDS IN SAMPLE FILE               136

FILE OUTFILE WILL BE CREATED
```

This report lists the input parameters followed by the results of the VARSAMP calculations. The report also provides the results of the random sampling process, including whether an output file is created.

VERSUS

The VERSUS routine creates a frequency distribution of one field versus another field as follows:

- The first field is the quantitative field of which value contributes to the statistical information recorded for each category.
- The second field is nonquantitative and acts as the categorizing field that determines the distribution of records in the report.
- The occurrence of each unique value in the nonquantitative field is tallied, and statistics regarding the associated quantitative field are accumulated for each category. A report with an optional graph is produced. The following information is provided:
  - Number of items per category
  - Percentage of quantitative field by category
  - Statistical mean and standard deviation by category and for the entire file
  - Minimum and maximum value in each category
  - Totals for count and percent

Syntax

```
%VERSUS1 infile field1 field2  {GRAPH   {percent asterisks}} [LRECL length]
                               {NOGRAPH                  }

%VERSUS2 [DBFILE infile]
```

`infile`

Specify the name of the input file to VERSUS. A valid name is any previously defined file.

`field1`

Specify the name of the quantitative field to be analyzed. The value in this field is used to accumulate statistical information for each category. A valid field name is any quantitative field defined in the input file. The value of field1 cannot be larger than 13 digits, including decimal places.

`field2`

Specify the name of the controlling field used to create the categories for accumulating the statistical information. When this field changes, a summary line with the statistical information from this category is printed on the report. The result is a report categorized by this field. A valid name is any nonquantitative field defined in the input file up to 20 characters in length.

```
{GRAPH  }
{NOGRAPH}
```

Specify whether you want a graph to be produced with the report.

GRAPH—Specifies that a graph is to be produced. When this option is selected, the percent and asterisks parameters must be coded.

NOGRAPH—Specifies that the graph is to be omitted. When this option is selected, do not code the following two parameters.

```
{percent}
```

If GRAPH is specified, use this parameter to define the occurrence percentage at which graphing can begin. The percentage value is subtracted from positive graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become less than zero are set equal to zero. The percentage value is added to negative graphing percentages to produce an adjusted graphing percentage. Adjusted percentages that become greater than zero are set equal to zero.

For example, if percent is 10, graphing percentages from –10 percent to +10 percent are displayed with no asterisks. A standard graph contains no adjustment and is produced by specifying a percentage value of zero. A valid value for percentage is an actual integer value greater than or equal to zero and less than 100, or the name of a field containing an integer value in the same range. Values with decimal places are truncated on the right.

```
{asterisks}
```

Graph lines are drawn with the asterisks character. If GRAPH is specified, use this parameter to define the number of asterisks that represent each percentage point. Valid values for asterisk include the actual numeric values 1 through 9 or the name of a field containing the values 1 through 9. For example, if an extremely flat distribution is anticipated, specify a higher value. This makes the graph easier to read. If a wide variance of percentages is anticipated, specify a lower value.

```
[LRECL length]
```

Optionally specify the length of the input record. The default is 32,767 bytes. If the record length is less than 32,767, you can improve the efficiency of both disk storage utilization and execution speed by specifying the exact length of the record according to the following formula:

```
Infile-lrecl + len-field1 + len-field2+ 1 work byte + 4 RDW bytes = LRECL
```

```
[DBFILE infile]
```

This optional parameter specifies a database file for use with VERSUS. Infile identifies the name of the input file to VERSUS. The name must be the same name that you specified for infile on the first invocation statement.

Operation — Stand-alone REPORT

Graphing

When using the percent and asterisk parameters in creating a graph, you must be careful to specify values that produce a meaningful graph. Many factors enter in to this evaluation, including the number of columns on the printer. In general:

- When the percentage of values to be graphed is large, give the asterisks parameter a low value (1 or 2).
- If a wide variance of percentages exist, specify a low value for asterisks.
- For graphing a narrow range of percentages, increase the asterisk value (3 through 9) for easier interpretation of results.
- If the graph overflows the print line, the letter O is printed between the PCT column and the graph, to indicate the overflow condition.
- The calculated percentages in the graph are the percentage of the amounts in each category, not the percentage of items in each category.

The VERSUS routine creates a CA-Easytrieve Plus table. Depending on the input data, the default allocation of 256 table entries may be exceeded. Error message A008 will inform you of this condition. For a further explanation of this message, see the CA-PanAudit Plus *Messages Guide*. To increase the allocation for table entries, the CA-Easytrieve Plus options table must be link edited with a new maximum value. For details, see the CA-Easytrieve Plus *Getting Started* guide.

Operation — Database

The DBFILE parameter identifies VERSUS as a routine that can access database files. This is an optional parameter that you need not specify for nondatabase use. However, you must specify this parameter when you use VERSUS in a database application.

Example

The following example of VERSUS gives a frequency distribution analysis of BILLING versus MONTH. The result is a report of amounts billed to customers for each month.

Input

```
FILE CUSTOMR FB (44 4400)
MONTH 1 2 N
BILLING 3 8 P 2
%VERSUS1 PERSNL GROSS MONTH GRAPH 0 2
%VERSUS2
```

Output

FREQUENCY DISTRIBUTION OF PAGE 1
GROSS VERSUS MONTH
INPUT FILENAME PERSNL

MONTH	TOTAL GROSS	COUNT	PCT	MEAN	STD DEV	MINIMUM	MAXIMUM
01	1,534.40	4	8.5	383.60	117.94	242.40	554.40
02	1,675.12	4	9.3	418.78	174.47	283.92	712.80
03	1,909.60	4	10.6	477.40	308.33	220.80	1,004.00
04	1,772.08	4	9.8	443.02	172.17	310.40	736.00
05	1,360.00	4	7.5	340.00	35.52	295.20	376.00
06	1,578.40	4	8.7	394.60	143.09	250.40	628.00
07	1,268.96	4	7.0	317.24	30.96	279.36	365.60
08	1,451.65	4	8.0	362.91	263.88	121.95	759.20
09	1,468.39	4	8.1	367.10	255.42	183.75	804.64
10	1,542.96	4	8.5	385.74	163.50	146.16	591.20
11	1,358.00	4	7.5	339.50	271.38	135.85	804.80
12	1,158.52	4	6.4	289.63	180.60	13.80	492.26
TOTALS	18,078.08	48	99.9	376.63	202.18	13.80	1,004.00

```
     FREQUENCY DISTRIBUTION GRAPH

                         PCT

 01        1,534.40    8.5      *****************
 02        1,675.12    9.3      *******************
 03        1,909.60   10.6      *********************
 04        1,772.08    9.8      ********************
 05        1,360.00    7.5      ***************
 06        1,578.40    8.7      *****************
 07        1,268.96    7.0      **************
 08        1,451.65    8.0      ****************
 09        1,468.39    8.1      ****************
 10        1,542.96    8.5      *****************
 11        1,358.00    7.5      ***************
 12        1,158.52    6.4      *************
```

This report shows that for each month, the total amount billed, count, percent, mean, standard deviation, minimum, and maximum values are listed. The percent listed is the percent of the amount in each category, not the percent of occurrences in each category. The graph starts at 0 percent, and two asterisks represent 1 percent.

WEEKDAY

The WEEKDAY routine calculates the day of the week from a given date. The date can be in any format. WEEKDAY only calculates the day of the week for the years 1940 through 2039. The day of the week calculated is written to a specified field.

Syntax

```
%WEEKDAY    date    format  day-of-week [THRESHOLD value]
```

`date`

Specify the name of the field containing the date for which the day of the week is calculated. The date in this field must be in the format specified by FORMAT. A valid name is any previously defined field.

`format`

Specify the format of the date field. Format is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The value of Date is not checked for a valid date with the specified format. If you want validation, use the DATEVAL routine before using WEEKDAY. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
```

`day-of-week`

Specify the name of a previously defined alphanumeric field to which the resulting day of the week is written. The field must be at least nine bytes to avoid truncating the day of the week.

`[THRESHOLD value]`

The THRESHOLD parameter is used to determine the century value if it is not supplied in the century format (CC) in the date. Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

Note: Unlike other macros that use the THRESHOLD value, the WEEKDAY routine has a default THRESHOLD value of 39. This provides a range from 1940 to 2039 inclusive. You can override this value.

General rules for specifying THRESHOLD values are:

- The THRESHOLD value is ignored if you provide a century value (CC).
- If the dates to be generated do not exceed the year 2000, specify the THRESHOLD a value of 0. This causes all dates to have a range of 1901 through 2000.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify the THRESHOLD default value of 0.
- Valid values for THRESHOLD are 0 through 99.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040, and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if WEEKDAY is invoked to process dates between the years 1949 and 1952, and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

Operation — Inline

WEEKDAY generates no output and can be used with other routines and/or CA-Easytrieve Plus logic. WEEKDAY does no date checking and assumes that the date input is valid.

Operation — Database

No change in specification of parameters is required to use WEEKDAY with database files.

Example

In this example, the day of the week is calculated from the date contained in the specified field, and the output is printed using a PRINT statement.

Input

```
FILE INFILE CARD
GREG-DATE    1 6 N HEADING ('GREGORIAN' 'DATE') MASK 'Z9/99/99'
*
DAY-OF-WEEK  W 9 A  HEADING ('DAY OF' 'WEEK')
*
JOB INPUT INFILE
   %WEEKDAY GREG-DATE MMDDYY DAY-OF-WEEK
   PRINT REPORT1
REPORT REPORT1 LINESIZE 78
   TITLE 1 'TEST OF DAY OF THE WEEK MACRO'
   *
   LINE GREG-DATE DAY-OF-WEEK
    *
 END
```

Output

```
4/13/88             TEST OF DAY OF THE WEEK MACRO            PAGE 1

                             GREGORIAN    DAY OF
                               DATE        WEEK

                             11/01/87    SUNDAY
                             11/02/87    MONDAY
                             11/03/87    TUESDAY
                             11/04/87    WEDNESDAY
                             11/05/87    THURSDAY
                             11/06/87    FRIDAY
                             11/07/87    SATURDAY
                             12/03/87    THURSDAY
                              1/04/87    SATURDAY
                              1/05/87    SUNDAY
                              2/06/87    THURSDAY
                              3/07/87    FRIDAY
                             11/08/87    SUNDAY
                             11/09/87    MONDAY
                             11/10/87    TUESDAY
                             11/11/87    WEDNESDAY
                             11/12/87    THURSDAY
                             11/13/87    FRIDAY
```

Chapter 9

Basic SMF Reporting Facility

The SMF reporting system provides a simple, yet flexible method to access and report on data from the IBM System Management Facilities (SMF). You can obtain results from the SMF reporting system by invoking macro statements that do not require a detailed knowledge of SMF records. The SMF reporting system provides two methods for accessing SMF data:

- **SMF Audit Routines** — These routines provide an easytouse method of accessing SMF data and formatting SMF data into meaningful reports.
- **SMF Record Field Definitions** — These routines provide the flexibility to support customized reports through user-written routines. These routines contain the field definitions for most SMF record types. For additional information, see SMF Record Field Definitions later in this chapter.

SMF Audit Routines

The SMF audit routines prepare various reports on the information in certain SMF record types. The following is a list of the SMF audit routine names and their functions:

Routine Name	SMF Record Type Reported On	Function Reported On
SMF000	00	IPLs
SMF004	04	Abnormal step terminations
SMF005	05	Abnormal job terminations
SMF006	06	Control output forms
SMF007	07	SMF lost data
SMF014	14	Data set activity
SMF017	17	Scratched data sets
SMF018	18	Renamed data sets
SMF020	20	Job initiations

Routine Name	SMF Record Type Reported On	Function Reported On
SMF049	49	JES2 and JES3 integrity (OS/390 and above only)
SMF062	62	VSAM opens
SMF067	67	VSAM data set deletes
SMF068	68	VSAM data sets renamed
SMFCNT	All SMF record types	Frequency distribution of record types showing record collection activities

SMF audit routines follow a naming convention of SMF0xx, where xx is the number of the SMF record type on which the SMF audit routine makes the report. The SMFCNT routine reports on all record types in a frequency distribution report. You can run SMFCNT first to determine the number and percentage of types of records that are on the SMF file.

You can invoke the SMF audit routines in a manner similar to stand-alone REPORT CA-PanAudit Plus routines. Each routine consists of two macro invocation statements and is invoked through the passing of parameters. In addition, you can stack routines to produce multiple reports from a single job step. You can also stack the same routine within one job.

By stacking routines you can generate multiple reports in one pass of the file. This increases the efficiency and flexibility of the reporting system. Each routine is separated into two parts:

- The first part performs the logic to collect the necessary data from the SMF file.
- The second part consists of the report to generate the listing.

To create the listing that you want, you must invoke both parts of a routine with the appropriate parameters.

The following example demonstrates the use of the SMF reporting system:

```
%SMFILE
%SMF014A DATE '''840101''' '''840131'''
%SMF014A JOBNAME '''A       ''' '''Z9999999''' REPORT J14#2
%SMF017A 000000 240000 840101 840131
%SMF018A 000000 240000 840101 840131
%SMF014B SORT TIME CONTROL DATE
%SMF014B SORT 'DSNAME TIME' CONTROL DATE REPORT J14#2
%SMF018B SORT 'DATE D'
%SMF017B SORT DATE
```

The first invocation is for the macro SMFILE. This macro contains the necessary file and field definitions to access the SMF data file. When using SMF audit routines, the first macro that you invoke must always be SMFILE. SMFILE must also be the DDNAME of the DD statement in the JCL referring to the SMF data file.

Note: For SMF files generated by MVS/XA 2.2.0, replace %SMFILE with %S22FILE to access the SMF data file. Likewise, for SMF files generated by MVS/ESA 3.1.3, replace %SMFILE with %SE13FILE to access the SMF data file.

This is followed by the invocations for the SMF routines. The first two invocations are for SMF014 reports. The selection criterion for the first report is the DATE field, and for the second report, the JOBNAME field. The first SMF014A routine is associated with the first SMF014B routine, which specifies sorting by the TIME field. The second SMF014A routine is associated with the second SMF014B routine by specifying J14#2 in the REPORT parameter of both SMF014A and SMF014B. When the same routine is invoked more than once in a single execution, the REPORT parameter associates the A and B routines.

Next, the SMF017 and SMF018 routines are invoked. Notice that SMF017A is invoked before SMF018A, while SMF018B is invoked before SMF017B. The D following the field DATE in the SORT parameter of SMF018B specifies the sorting of records in descending sequence.

The rules for specifying the A and B routines are:

- All A routines must precede all B routines.
- The reports generate in the order of specification of the B routines.
- The REPORT parameter associates the processing logic (A routine) with the actual report (B routine).
- You can insert screening code between the SMFILE statement and the first A routine. However, screening applies to all reports in the processing stream.

SMF Record Field Definitions

These routines provide field definitions for the majority of SMF record types. The field names follow the conventions listed in the IBM guide *OS/390 MVS System Management Facilities (SMF)*. These routines provide the field definitions for customized SMF reports that you write in CA-Easytrieve Plus . These routines eliminate the time-consuming task of coding field definitions for the various SMF record types.

The following example illustrates accessing an SMF data file:

```
FILE SMFDATA ...
%SMFR20
...user written code ....
```

The SMFR20 routine provides the definitions for the various fields of a pre-MVS 2.2.0 SMF Type 20 record. You can then write CA-Easytrieve Plus statements that See any field in a type 20 SMF record. Since many fields in an SMF data record are variable in length, indexing methods are used to access these fields. Later in this chapter, you can find a description of the operation of each SMF record type.

IPL Reporting — SMF000

The SMF000 routine creates a report listing each occurrence of an Initial Program Load (IPL) by the CPU, the time and date of each IPL, and the system options in effect. The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF000A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF000B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on system IPLs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on system IPLs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on system IPLs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on system IPLs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

```
[REPORT reportname]
```

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical report name must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF000.

Input

```
%SMFILE
%SMF000A 000000 240000 850318 850323
%SMF000B
```

Output

```
        SMF000 - INITIAL PROGRAM LOAD FOR SYSTEM 3083


                                            SMF          NUMBER
            IPL          IPL              ACCOUNT          OF
            DATE         TIME             REQUESTS         IPLS

          85/03/19     17:05:16        SYSTEM.JOB
                                       SYSTEM.JOB.STEP
                                       USER EXITS
                                       DATA SET
                                       VOLUME
                                       TEMP DATA SET

          85/03/20      1:53:22        SYSTEM.JOB
                                       SYSTEM.JOB.STEP
                                       USER EXITS
                                       DATA SET
                                       VOLUME
                                       TEMP DATA SET

          85/03/22      5:26:30        SYSTEM.JOB
                                       SYSTEM.JOB.STEP
                                       USER EXITS
                                       DATA SET
                                       VOLUME
                                       TEMP DATA SET

          FINAL TOTAL                                        3
```

This report lists all IPL reports in the SMF data set.

Abnormal Step Termination — SMF004

The SMF004 routine creates a report of abnormally terminated steps and computes the elapsed time for the step. The parameters that you pass allow for selection, sorting, and totaling based on DATE, TIME, JOBNAME, PROGRAM_, USERID_, or ABEND.

Syntax

```
%SMF004A selection {selectstart selectend                      }
                   {'''STRING''' '''value1,value2,...valuen'''}  +

                   [REPORT reportname]

%SMF004B [REPORT reportname] [SORT sortflds [D]]   +

         [CONTROL cntrlflds [options]]
```

Specify the field in the SMF record that determines the range for selection of records in this report. The specification of this parameter determines the valid values for the selectstart and selectend parameters. The value that you specify for selection must be one of the following:

DATE—Date determines selection. Selectstart, selectend, or the individual values that STRING specifies must be dates in YYMMDD format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual six-digit numeric values.

TIME—Time determines selection. Selectstart, selectend, or the individual values that STRING specifies must be times in HHMMSS format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual six-digit numeric values.

JOBNAME—Jobname determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing job names. Valid values for selectstart and selectend are an eight-byte alphanumeric field or a character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

USERID_—User ID determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing user IDs. Valid values for selectstart and selectend are an eight-byte alphanumeric field or a character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

PROGRAM_—Program name determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing program names. Valid values for selectstart and selectend are an eight-byte alphanumeric field or character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

ABEND—Abend code determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing abend codes. You must define selectstart and selectend as twobyte binary fields containing valid abend values, such as 00C4, 00C7, etc. Numeric values for selectstart and selectend are not allowed when ABEND is specified in the selection parameter.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. The values must be actual alphanumeric values representing valid four-digit hexadecimal numbers.

```
{selectstart selectend                  }
{'''STRING''' '''value1,value2,...valuen'''}
```

Specify whether selection is to be by starting and ending points or by a series of individual values.

`selectstart`

Specify the beginning of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

`selectend`

Specify the end of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

`STRING`

Specify a series of individual values, separated by commas. The keyword STRING and the string of individual values must each be enclosed in triple quotes. The total length of the string, including commas and triple quotes, must not be greater than 254 characters.

Examples

```
%SMF004A DATE '''STRING''' '''860612,861130,870101,871231,880101'''
%SMF004A ABEND '''STRING''' '''0322,00C4,00C7,0806'''
%SMF004A JOBNAME '''STRING''' '''YEAREND,ARDEPT03,PAYROLL1,AUDIT'''
```

These values are the individual values that are reported on for the particular select parameter. Valid values vary depending upon the field chosen for the selection parameter.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

`[SORT sortflds [D]`

Specify the field or fields that you want to use to sequence the report. Valid values are any of the field names described in the selection parameter. To specify multiple fields for sequencing, enclose the field names in single quotes and separate the field names by one blank space.

To specify a descending sort sequence, insert the letter D after the appropriate field name. You must use a blank space to separate the D from the previous and any subsequent field names. When you specify descending sequence, enclose all items in the parameter list in single quotes.

`[CONTROL cntrlflds [options]]`

Specify the field or fields that you want for control breaks in the reporting process. Valid values for cntrlflds are any of the field names described in the selection parameter. To specify multiple fields for control breaks, enclose the field names in single quotes and separate them by one blank space.

You can specify report processing options after the control break field. These options customize the report in relation to control break activities. The options are:

NEWPAGE—Causes a skip to top-of-page after processing is complete for the control break field.

RENUM—Performs the same function as NEWPAGE and also resets the page number to one following the control break.

NOPRINT—Suppresses printing of the summary line group for the control break that you specify.

If you specify any of these options, separate all items in the parameter list by one blank space and enclose the entire string in single quotes.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Examples

The following are two examples of SMF004.

Example One

Input

```
%SMFILE
DEFINE ABEND-SELECT   W   2   B   VALUE(X'0322')
%SMF004A ABEND ABEND-SELECT ABEND-SELECT
%SMF004B SORT 'PROGRAM_ DATE TIME' CONTROL PROGRAM
```

Output

```
                   SMF004 - ABNORMAL STEP TERMINATION REPORT                     PAGE       1
                                                  SYSTEM ID 3081
                                                SELECTED BY ABEND
                                      SEQUENCED BY SYSID PROGRAM DATE TIME
                                                                         MINUTES
           COMPLETION  ABEND                                             ELAPSED
              CODE       TYPE    JOBNAME   DATE       TIME    USERID  PROGRAM   TIME       TALLY

              0322     SYSTEM    JONES    89/12/19 15:40:40-          JIFSEL    11.3685
           PROGRAM TOTAL                                                        11.3685      1

           SYSID TOTAL                                                          11.3685      1
           FINAL TOTAL                                                          11.3685      1

TOTAL TYPE 4 RECORDS              1,439
TOTAL SELECTED                        1
PERCENT                                .06
```

This report lists all 0322 abnormal terminations in the SMF data set. The desired ABEND code must be passed to the routine in a previously defined field. ABEND-SELECT is defined as a twobyte binary field containing the value '0322' and is specified as the SELECTSTART and SELECTEND parameters in the SMF004A invocation statement. The report is sequenced by PROGRAM_, DATE, and TIME within system id (SYSID).

Example Two

Input

```
%SMFILE
%SMF004A ABEND '''STRING''' '''00C4,0213,0222'''
%SMF004B SORT 'PROGRAM_ DATE TIME' CONTROL PROGRAM
```

Output

```
                      SMF004 - ABNORMAL STEP TERMINATION REPORT                         PAGE 1
                                        SYSTEM ID 3081
                                      SELECTED BY ABEND
                            SEQUENCED BY SYSID PROGRAM DATE TIME
                                                                                 MINUTES
             COMPLETION  ABEND                                                   ELAPSED
                CODE      TYPE   JOBNAME    DATE      TIME   USERID  PROGRAM      TIME    TALLY

                0222    SYSTEM   SMITH     89/12/19 14:39:31-        PSIS5320   3.4201
                0222    SYSTEM   SMITH     89/12/19 14:49:32-        PSIS5320   4.5176
                0222    SYSTEM   SMITH     89/12/19 15:50:30-        PSIS5320   3.3060
             PROGRAM TOTAL                                                     11.2437      3

                0222    SYSTEM   DBMSPCAT  89/12/19 15:59:27-        FDRDSF     1.1030
             PROGRAM TOTAL                                                      1.1030      1

                0222    SYSTEM   JONES     89/12/19 11:27:15-        JIFSEL     5.2191
             PROGRAM TOTAL                                                      5.2191      1

                0222    SYSTEM   KINSSCAN  89/12/19  9:00:53-        TAPESCAN  26.7758
             PROGRAM TOTAL                                                     26.7758      1

               SYSID TOTAL                                                     44.3416      6
               FINAL TOTAL                                                     44.3416      6

TOTAL TYPE 4 RECORDS           1,439
TOTAL SELECTED                     6
PERCENT                              .41
```

This sample lists 00C4, 0213, and 0222 abnormal terminations in the SMF data set. The desired ABEND codes are passed to the SMF004A routine by means of the STRING option and a string of abend codes. The report is sequenced by PROGRAM_, DATE, and TIME within system id (SYSID).

Abnormal Job Terminations — SMF005

The SMF005 routine creates a report of jobs that terminate abnormally and computes the elapsed time for the job. The parameters that you pass allow for selection, sorting, and totaling based on DATE, TIME, JOBNAME, PROGRAMMER, USERID_, or ABEND.

Syntax

```
%SMF005A selection {selectstart  selectend                 }
                   {'''STRING''' '''value1,value2,...valuen'''} +

                   [REPORT reportname]

%SMF005B [REPORT reportname] [SORT sortflds [D]]  +

         [CONTROL cntrlflds [options]]
```

`selection`

Specify the field in the SMF record that determines the range for selection of records in this report. The specification of this parameter determines the valid values for the selectstart and selectend parameters. The value that you specify for selection must be one of the following:

DATE—Date determines selection. Selectstart, selectend, or the individual values that STRING specifies must be dates in YYMMDD format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual six-digit numeric values.

TIME—Time determines selection. Selectstart, selectend, or the individual values that STRING specifies must be times in HHMMSS format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual six-digit numeric values.

JOBNAME—Jobname determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing job names. Valid values for selectstart and selectend are an eight-byte alphanumeric field or a character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

USERID_—User ID determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing user IDs. Valid values for selectstart and selectend are an eight-byte alphanumeric field or a character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

PROGRAM_—Program name determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing program names. Valid values for selectstart and selectend are an eight-byte alphanumeric field or character string up to eight characters long enclosed in triple quotes.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. Valid values are actual alphanumeric values, each containing a character string up to eight characters long.

ABEND—Abend code determines selection. Selectstart, selectend, or the individual values that STRING specifies must be alphabetic literals representing abend codes. You must define selectstart and selectend as twobyte binary fields containing valid abend values, such as 00C4, 00C7, and so on. Numeric values for selectstart and selectend are not allowed when ABEND is specified in the selection parameter.

If using the STRING option, the keyword STRING and the string of individual values must each be enclosed in triple quotes. Separate each individual value in the string with a comma. The values must be actual alphanumeric values representing valid four-digit hexadecimal numbers.

```
{selectstart selectend                   }
{'''STRING''' '''value1,value2,...valuen'''}
```

Specify whether selection is to be by starting and ending points or by a series of individual values.

`selectstart`

Specify the beginning of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

`selectend`

Specify the end of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

STRING

Specify a series of individual values, separated by commas. The keyword STRING and the string of individual values must each be enclosed in triple quotes. The total length of the string, including commas and triple quotes, must not be greater than 254 characters.

Examples

```
%SMF005A DATE '''STRING''' '''860612,861130,870101,871231,880101'''
%SMF005A ABEND '''STRING''' '''0322,00C4,00C7,0806'''
%SMF005A JOBNAME '''STRING''' '''YEAREND,ARDEPT03,PAYROLL1,AUDIT'''
```

[REPORT reportname]

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

[SORT sortflds [D]]

Specify the field or fields that you want to use to sequence the report. Valid values are any of the field names described in the selection parameter. To specify multiple fields for sequencing, enclose the field names in single quotes and separate the field names by one blank space.

To specify a descending sort sequence, insert the letter D after the appropriate field name. You must use a blank space to separate the D from the previous and any subsequent field names. When you specify descending sequence, enclose all items in the parameter list in single quotes.

[CONTROL cntrlflds [options]]

Specify the field or fields that you want for control breaks in the reporting process. Valid values for cntrlflds are any of the field names described in the selection parameter. To specify multiple fields for control breaks, enclose the field names in single quotes and separate them by one blank space.

You can specify report processing options after the control break field. These options customize the report in relation to control break activities. These options are:

NEWPAGE—Causes a skip to top-of-page after processing is complete for the control break field.

RENUM—Performs the same function as NEWPAGE and also resets the page number to one following the control break.

NOPRINT—Suppresses printing of the summary line group for the control break that you specify.

If you specify any of these options, separate all items in the parameter list by one blank space and enclose the entire string in single quotes.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Examples

The following are two examples of SMF005.

Example One

Input

```
%SMFILE
%SMF005A DATE '''891219''' '''891219'''
%SMF005B SORT 'PROGRAMMER DATE TIME' CONTROL PROGRAMMER
```

Output

```
                    SMF005 - ABNORMAL JOB TERMINATION REPORT                              PAGE 1
                                    SYSTEM ID 3081
                                   SELECTED BY DATE
                        SEQUENCED BY SYSID PROGRAMMER DATE TIME

                                                                            MINUTES
COMPLETION  ABEND                                        PROGRAMMER         ELAPSED      ACCOUNT
   CODE      TYPE   JOBNAME    DATE      TIME    USERID       NAME            TIME       NUMBER   TALLY

   0013    SYSTEM   JOB       89/12/19  7:25:33                               .0285
   01F6    USER     ABEL      89/12/19 13:39:02 -                             .4793       626B
   0013    SYSTEM   JOB       89/12/19 14:52:57                               .0203
PROGRAMMER TOTAL                                                              .5281

   0222    SYSTEM   SMITH1    89/12/19 14:39:31 -         SMITH              3.7870       606A
   0222    SYSTEM   SMITH1    89/12/19 14:49:32 -         SMITH              5.1045       606A
   0222    SYSTEM   SMITH1    89/12/19 15:50:30 -         SMITH              3.5668       606A
PROGRAMMER TOTAL                                                            12.4583

   0222    SYSTEM   DBMSPCAT  89/12/19 15:59:27 -         DATA CENTER BKUPS  1.1035       625A
   0378    USER     DBUSR034  89/12/19 22:21:59 -         DATA CENTER BKUPS 28.0801       625A
   0378    USER     DBUSR035  89/12/19 22:37:33 -         DATA CENTER BKUPS 29.7345       625A
   0378    USER     DBUSR051  89/12/19 23:13:53 -         DATA CENTER BKUPS 27.3636       625A
PROGRAMMER TOTAL                                                            86.2817

   0806    SYSTEM   JOHNSON   89/12/19 10:19:51 -         JIFSEL              .3183       609A
   0222    SYSTEM   JOHNSON   89/12/19 11:27:15 -         JIFSEL             5.2193       609A
   0322    SYSTEM   JOHNSON   89/12/19 15:40:40 -         JIFSEL            11.3690       609A
PROGRAMMER TOTAL                                                            16.9066

   0222    SYSTEM   JONE      89/12/19  9:00:53 -         JONES             26.7761       630A
PROGRAMMER TOTAL                                                            26.7761

   0706    SYSTEM   JACKSON1  89/12/19 15:36:50 -         JACKSON             .3045       606A
   00C1    SYSTEM   JACKSON1  89/12/19 17:33:38 -         JACKSON             .1711       606A
   00C1    SYSTEM   JACKSON1  89/12/19 17:35:54 -         JACKSON             .1578       606A
PROGRAMMER TOTAL                                                              .6334

SYSID TOTAL                                                                143.5842
FINAL TOTAL                                                                143.5842


TOTAL TYPE 5 RECORDS             466
TOTAL SELECTED                    17
PERCENT                            3.64
```

This sample lists all abnormal terminations of jobs in the SMF data set. The report is sequenced by PROGRAMMER, DATE, and TIME with control breaks on the field PROGRAMMER within system id (SYSID).

Example Two

Input

```
%SMFILE
%SMF005A DATE '''STRING''' '''891219'''
%SMF005B SORT 'PROGRAMMER DATE TIME' CONTROL PROGRAMMER
```

Output

```
                     SMF005 - ABNORMAL JOB TERMINATION REPORT                              PAGE 1
                                  SYSTEM ID 3081
                                 SELECTED BY DATE
                       SEQUENCED BY SYSID PROGRAMMER DATE TIME

                                                                          MINUTES
COMPLETION  ABEND                                             PROGRAMMER     ELAPSED      ACCOUNT
   CODE       TYPE   JOBNAME    DATE      TIME    USERID          NAME          TIME       NUMBER   TALLY

   0013     SYSTEM   JOB       89/12/19  7:25:33                                 .0285
   01F6     USER     ABEL      89/12/19 13:39:02 -                               .4793        626B
   0013     SYSTEM   JOB       89/12/19 14:52:57                                 .0203
PROGRAMMER TOTAL                                                                 .5281

   0222     SYSTEM   SMITH1    89/12/19 14:39:31 -         SMITH                3.7870        606A
   0222     SYSTEM   SMITH1    89/12/19 14:49:32 -         SMITH                5.1045        606A
   0222     SYSTEM   SMITH1    89/12/19 15:50:30 -         SMITH                3.5668        606A
PROGRAMMER TOTAL                                                               12.4583

   0222     SYSTEM   DBMSPCAT  89/12/19 15:59:27 -         DATA CENTER BKUPS    1.1035        625A
   0378     USER     DBUSR034  89/12/19 22:21:59 -         DATA CENTER BKUPS   28.0801        625A
   0378     USER     DBUSR035  89/12/19 22:37:33 -         DATA CENTER BKUPS   29.7345        625A
PROGRAMMER TOTAL                                                               58.9181

   0806     SYSTEM   JOHNSON   89/12/19 10:19:51 -         JIFSEL                .3183        609A
   0222     SYSTEM   JOHNSON   89/12/19 11:27:15 -         JIFSEL               5.2193        609A
   0322     SYSTEM   JOHNSON   89/12/19 15:40:40 -         JIFSEL              11.3690        609A
PROGRAMMER TOTAL                                                               16.9066

   0222     SYSTEM   JONE      89/12/19  9:00:53 -         JONES               26.7761        630A
PROGRAMMER TOTAL                                                               26.7761

   0706     SYSTEM   JACKSON1  89/12/19 15:36:50 -         JACKSON               .3045        606A
   00C1     SYSTEM   JACKSON1  89/12/19 17:33:38 -         JACKSON               .1711        606A
   00C1     SYSTEM   JACKSON1  89/12/19 17:35:54 -         JACKOSN               .1578        606A
PROGRAMMER TOTAL                                                                 .6334

SYSID TOTAL                                                                   116.2206
FINAL TOTAL                                                                   116.2206

TOTAL TYPE 5 RECORDS             466
TOTAL SELECTED                    16
PERCENT                            3.43
```

This sample lists all abnormal terminations of jobs in the SMF data set. The date is passed to the SMF005A routine by means of the STRING option. The report is sequenced by PROGRAMMER, DATE, and TIME with control breaks on the field PROGRAMMER within system id (SYSID).

Control Output Forms — SMF006

The SMF006 routine creates a report listing where printed reports are routed and if any operator intervention occurred. The report is sequenced by TIME, DATE, JOBNAME, and FORM#. Control breaks occur when the FORM# field changes within system id (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF006A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF006B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on output forms. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on output forms. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on output forms. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on output forms. A valid value is either an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF006.

Input

```
%SMFILE
%SMF006A 000000 240000 891219 891220
%SMF006B
```

Output

```
                       SMF006 - OUTPUT WRITER FORMS CONTROL REPORT                                  PAGE    1
                                          SYSTEM 3081
                                           FORM BLNK

                                       FORM           RECORD                      OPERATOR           JOB
JOBNAME        DATE          TIME     NUMBER           COUNT          ROUTED    INTERVENTION         COUNT

PR90000P     89/12/19      1:09:41     BLNK                 1,593
PR90000P     89/12/19     14:26:42     BLNK                 1,539                  RESTARTED
FORM# TOTAL                                                 3,132                                       2

                           SMF006 - OUTPUT WRITER FORMS CONTROL REPORT                                  PAGE    2
                                           SYSTEM 3081
                                            FORM JMTL

                                        FORM           RECORD                     OPERATOR           JOB
JOBNAME        DATE          TIME      NUMBER           COUNT         ROUTED    INTERVENTION         COUNT

OS10JEW      89/12/19     13:34:09      JMTL                  135
OS10JRN      89/12/20     10:31:19      JMTL                  144
OS10JRN      89/12/19     14:26:41      JMTL                  225
OS10JRW      89/12/20     14:26:43      JMTL                1,152
FORM# TOTAL                                                 1,656                                       4

SYSID TOTAL                                                 4,788                                       6

FINAL TOTAL                                                 4,788                                       6
```

This sample lists output forms control information in the SMF data set.

SMF Data Lost — SMF007

The SMF007 routine creates a report listing the date, time, elapsed time, and number of records lost in any interruption of SMF recording. The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF007A starttime endtime startdate enddate [REPORT reportname]

%SMF007B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on lost SMF data. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on lost SMF data. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on lost SMF data. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on lost SMF data. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF007.

Input

```
%SMFILE
%SMF007A 130000 140000 890426 890426
%SMF007B
```

Output

```
                SMF007 - DATA LOST (SMF NOT RECORDED) REPORT
                                SYSTEM 3083


                                       MINUTES
                                       ELAPSED              NUMBER OF
      DATE          TIME                TIME               RECORDS LOST

    89/04/26       13:51:37                .0483                 11

    89/04/26       13:51:38                .4166                 47

    89/04/26       13:51:40                .7183                 19

 SYSID TOTAL                              1.1832

 FINAL TOTAL                              1.1832
```

This sample lists all occurrences of lost SMF data in the data set.

Data Set Activity — SMF014

The SMF014 routine creates a report of data set activity including the accessing jobname, date, and time. The parameters that you pass allow for selection, sorting, and totaling based on DATE, TIME, data set name (DSNAME), or JOBNAME.

Syntax

```
%SMF014A selection selectstart selectend [REPORT reportname]

%SMF014B [REPORT reportname] [SORT sortflds [D]]  +

        [CONTROL cntrlflds [options]]
```

`selection`

Specify the field in the SMF record that determines the range for selection of records in this report. The specification of this parameter determines the valid values for the selectstart and selectend parameters. The value that you specify for selection must be one of the following:

DATE—Date determines selection. The selection range that selectstart and selectend specify must be dates in YYMMDD format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

TIME—Time determines selection. The selection range that selectstart and selectend specify must be times in HHMMSS format. Valid values for selectstart and selectend are a six-byte alphanumeric field or actual six-digit numeric values enclosed in triple quotes.

DSNAME—Data set name determines selection. The selection range that selectstart and selectend specify must be alphabetic literals representing data set names. Valid values for selectstart and selectend are a 44-byte alphanumeric field or a character string up to 44 characters long enclosed in triple quotes.

JOBNAME—Jobname determines selection. The selection range that selectstart and selectend specify must be alphabetic literals representing job names. Valid values for selectstart and selectend are an eight-byte alphanumeric field or a character string up to eight bytes long enclosed in triple quotes.

`selectstart`

Specify the beginning of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

`selectend`

Specify the end of the selection range for the field chosen in the selection parameter. Valid values vary depending on the field chosen for the selection parameter.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

`[SORT sortflds [D]]`

Specify the field or fields that you want to use to sequence the report. Valid values are any of the field names described in the selection parameter. To specify multiple fields for sequencing, enclose the field names in single quotes and separate the field names by one blank space.

To specify a descending sort sequence, insert the letter D after the appropriate field name. You must use a blank space to separate the D from the previous and any subsequent field names. When you specify descending sequence, enclose all items in the parameter list in single quotes.

`[CONTROL cntrlflds [options]]`

Specify the field or fields that you want for control breaks in the reporting process. Valid values for cntrlflds are any of the field names described in the selection parameter. To specify multiple fields for control breaks, enclose the field names in single quotes and separate them by one blank space.

You can specify report processing options after the control break field. These options customize the report in relation to control break activities. These options are:

NEWPAGE—Causes a skip to top-of-page after processing is complete for the control break field.

RENUM—Performs the same function as NEWPAGE and also resets the page number to one following the control break.

NOPRINT—Suppresses printing of the summary line group for the control break that you specify.

If you specify any of these options, separate all items in the parameter list by one blank space and enclose the entire string in single quotes.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF014.

Input

```
%SMFILE
%SMF014A TIME '''080000''' '''090000'''
%SMF014B SORT 'PROGRAM_ TIME' CONTROL PROGRAM
```

Output

```
                                    SMF014 - DATA SET ACTIVITY REPORT
                                            SYSTEM ID 3081
                                           SELECTED BY TIME
                                     SEQUENCED BY SYSID PROGRAM TIME

                               DATA SET                                          NUMBER OF
JOBNAME                          NAME                   DATE          TIME      DATASETS

LINK1          SYSTEM.LOADLIB                         89/03/20      8:21:18
LINK1          SYSTEM.LOADLIB                         89/03/20      8:21:19
PROGRAM TOTAL                                                                          2

NET            SYS1.VTAMLST                           89/03/20      8:42:24
NET            SYS1.VTAMLST                           89/03/20      8:42:53
NET            SYS1.VTAMLST                           89/03/20      8:43:27
PROGRAM TOTAL                                                                          3

 .                  .                                .         .           .
 .                  .                                .         .           .
 .                  .                                .         .           .
PAN1           SYSTEM.PANLIB                          89/03/20      8:16:52
PROGRAM TOTAL                                                                          2

PAN2           SYSTEM.PANLIB                          89/03/20      8:24:48
PAN2           SYSTEM.PANLIB                          89/03/20      8:24:51
PAN2           SYSTEM.PANLIB                          89/03/20      8:25:00
PROGRAM TOTAL                                                                          3

SMITH          SMITH.TESTFILE                         89/03/20      8:49:16
SMITH          SMITH.TESTFILE                         89/03/20      8:54:16
SMITH          SMITH.TESTFILE                         89/03/20      8:57:36
PROGRAM TOTAL                                                                          3

TOPPER         SYSTEM.EZTVFM                          89/03/20      8:25:35
TOPPER         SYSTEM.EZTVFM                          89/03/20      8:25:35
PROGRAM TOTAL                                                                          2

TOPPER2        SYSTEM.EZTVFM                          89/03/20      8:49:17
TOPPER2        SYSTEM.EZTVFM                          89/03/20      8:49:17
PROGRAM TOTAL                                                                          2
```

```
TEST1         SYSTEM.TEST.LOAD                      89/03/20      8:35:15
PROGRAM TOTAL                                                             1

SYSID TOTAL                                                             550

FINAL TOTAL                                                             550
```

This sample lists all data set activity for all days present in the SMF data set. The report is sequenced by TIME and PROGRAM_ with control breaks on the field PROGRAM_ within system id (SYSID).

Scratched Data Sets — SMF017

The SMF017 routine creates a report listing scratched data sets. A field that you specify within system ID (SYSID) sequences the report. The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF017A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF017B [REPORT reportname] [SORT sortflds [D]]
```

`starttime`

Specify the time that the routine will begin reporting on scratched data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on scratched data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on scratched data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on scratched data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

```
[REPORT reportname]
```

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

```
[SORT sortflds [D]
```

Specify the field or fields that you want to use to sequence the report. The value for sortfld must be one of the following:

- DATE
- TIME
- DSNAME (data set name)
- JOBNAME
- USERID_

If you want to specify multiple fields for sequencing, enclose the field names in single quotes and separate the field names by one blank space.

To specify a descending sort sequence, insert the letter D after the appropriate field name. You must use a blank space to separate the D from the previous and any subsequent field names. When you specify descending sequence, enclose all items in the parameter list in single quotes.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF017.

Input

```
%SMFILE
%SMF017A 000000 080000 891219 891219
%SMF017B SORT 'JOBNAME TIME'
```

Output

```
                     SMF017 - SCRATCHED DATA SET REPORT                    PAGE     1
                                SYSTEM 3081
                       SEQUENCED BY SYSID JOBNAME TIME

               DATA SET                                                        NUMBER OF
                 NAME                  JOBNAME      DATE       TIME    USERID   VOLUMES

SMITH.APT.R012ABS.T.APTMDLO             SMITH      89/12/19   7:40:01                1
SYS89353.T074258.RA000.SMITH.R0000003   SMITH      89/12/19   7:40:01                1
SYS89353.T074300.RA000.SMITH.R0000004   SMITH      89/12/19   7:40:01                1
SYS89353.T074301.RA000.SMITH.R0000005   SMITH      89/12/19   7:40:01                1
SYS89353.T074303.RA000.SMITH.R0000006   SMITH      89/12/19   7:40:01                1
SYS89353.T074304.RA000.SMITH.R0000007   SMITH      89/12/19   7:40:01                1
SYS89353.T074306.RA000.SMITH.R0000008   SMITH      89/12/19   7:40:01                1
SMITH.APT.R012ABS.T.APTMDLO             SMITH      89/12/19   7:40:01                1
SMITH.SPFLOG4.LIST                      SMITH      89/12/19   7:40:01                1
SMITH.SPFLOG4.LIST                      SMITH      89/12/19   7:40:01                1
SYS89353.T072234.RA000.RYAN.R0000001    RYAN       89/12/19   7:22:32                1
SYS89353.T072234.RA000.RYAN.R0000002    RYAN       89/12/19   7:22:32                1
SYS89353.T072234.RA000.RYAN.R0000003    RYAN       89/12/19   7:22:32                1
SYS89353.T072234.RA000.RYAN.R0000004    RYAN       89/12/19   7:22:32                1
SYS89353.T072235.RA000.RYAN2.R0000001   RYAN2      89/12/19   7:22:33                1
SYS89353.T072235.RA000.RYAN2.R0000002   RYAN2      89/12/19   7:22:33                1
SYS89353.T072235.RA000.RYAN2.R0000003   RYAN2      89/12/19   7:22:33                1
SYS89353.T072235.RA000.RYAN2.R0000004   RYAN2      89/12/19   7:22:33                1
RYAN2.SPFLOG1.LIST                      RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.BROWSE2         RYAN2      89/12/19   7:22:33                1
RYAN2.SPFLOG1.LIST                      RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.BROWSE2         RYAN2      89/12/19   7:22:33                1
RYAN2.SPFLOG1.LIST                      RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.BROWSE2         RYAN2      89/12/19   7:22:33                1
RYAN2.TESTDATA.ISPFOPTN.EDIT001         RYAN2      89/12/19   7:22:33                1
JACKSON.SPFTEMP0.CNTL                   JACKSON    89/12/19   7:31:08                1
JACKSON.PANVALET.ISPFOPTN.EDIT002       JACKSON    89/12/19   7:31:08                1
SYS89353.T073108.RA000.JACKSON.R0000003 JACKSON    89/12/19   7:31:08                1
SYS89353.T073108.RA000.JACKSON.R0000004 JACKSON    89/12/19   7:31:08                1
SYS89353.T073527.RA000.JACKSON.R0000044 JACKSON    89/12/19   7:31:08                1
SYS89353.T073531.RA000.JACKSON.R0000045 JACKSON    89/12/19   7:31:08                1
SYS89353.T073108.RA000.JACKSON.R0000002 PEARSON    89/12/19   7:31:08                1
SYS89353.T073108.RA000.JACKSON.R0000001 JACKSON    89/12/19   7:31:08                1
SYS89353.T074026.RA000.JACKSONC.R0000001 JACKSONC  89/12/19   7:40:25                1
SYS89353.T074026.RA000.JACKSONC.R0000002 JACKSONC  89/12/19   7:40:25                1
SYS89353.T074112.RA000.JACKSONC.R0000001 JACKSONC  89/12/19   7:40:37                1
SYS89353.T074112.RA000.JACKSONC.R0000002 JACKSONC  89/12/19   7:40:37                1
```

This sample lists data sets scratched in the SMF data set. The report is sequenced by data set name, TIME, and JOBNAME within SYSID.

Renamed Data Sets — SMF018

The SMF018 routine creates a report listing renamed data sets. The report is sequenced by an optionally specified field within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF018A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF018B [REPORT reportname] [SORT sortflds [D]]
```

`starttime`

Specify the time that the routine will begin reporting on renamed data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on renamed data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on renamed data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on renamed data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

```
[SORT sortflds [D]
```

Specify the field that you want to use to sequence the report. The value for sortfld must be one of the following:

- DATE
- TIME
- JOBNAME
- NEWNAME (new data set name)
- OLDNAME (old data set name)

To specify multiple fields for sequencing, enclose the field names in single quotes and separate the field names by one blank space.

To specify a descending sort sequence, insert the letter D after the appropriate field name. You must use a blank space to separate the D from the previous and any subsequent field names. When you specify descending sequence, enclose all items in the parameter list in single quotes.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF018.

Input

```
%SMFILE
%SMF018A 000000 240000 891219 891220
%SMF018B SORT 'JOBNAME TIME'
```

Output

```
                                SMF018 - RENAMED DATA SET REPORT                             PAGE      1
                                         SYSTEM 3081
                                SEQUENCED BY SYSID JOBNAME TIME

                OLD DATA SET                NEW DATA SET                                               VOL
                    NAME                        NAME                 JOBNAME    DATE      TIME   USERID  CNT

PSISYS.MINIDISK.RETSVML.D195      PSISYS.OLD.MINIDISK.RETSVML.D195   DIRADREN 89/12/19 10:14:21  -        1
PSISYS.MINIDISK.SPARE03           PSISYS.MINIDISK.RETSVML.D195       DIRADREN 89/12/19 10:14:23  -        1
TECHS.OLD.MINIDISK.BROWNIN        TECHS.MINIDISK.CC255.YURKOVI       DIRADREN 89/12/20 16:46:07  -        1
JACKSON.P1032858.PANDD3           JACKSON.P1032858.PANDD3            JACKSON  89/12/19 10:50:16  -        1
```

This sample lists data sets renamed at any time of day in the SMF data set. The report is sequenced by TIME and JOBNAME within SYSID.

Job Initiations — SMF020

The SMF020 routine creates a report listing jobs initiated by date, time, and programmer name. The report is sequenced by TIME, DATE, and JOBNAME within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF020A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF020B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on initiated jobs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on initiated jobs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on initiated jobs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on initiated jobs. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF020.

Input

```
%SMFILE
%SMF020A 000000 070000 891219 891219
%SMF020B
```

Output

```
              SMF020 - JOB INITIATION REPORT                                 PAGE      1
                         SYSTEM 3081

                                              PROGRAMMER      NUM OF
 JOBNAME     DATE      TIME    USERID            NAME          JOBS

 DBUSR054 89/12/19  0:01:26  -         DATA.CENTER BKUPS
                                                                 1
 DEALLOC  89/12/19  0:04:01
 DEALLOC  89/12/19  0:18:47
                                                                 2
 JOB      89/12/19  0:00:03
 JOB      89/12/19  0:01:24
                                                                 2
```

This sample lists jobs initiated in the SMF data set.

JES2 and JES3 Integrity — SMF049

The SMF049 routine creates a report listing occurrences of password invalidity and reason for failure of RJE station signon. The report is sequenced by TIME, DATE, and LINE within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF049A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF049B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on JES2 and JES3 integrity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on JES2 and JES3 integrity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on JES2 and JES3 integrity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on JES2 and JES3 integrity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF049.

Input

```
%SMFILE
%SMF049A 130000 140000 890426 890426
%SMF049B
```

Output

```
                                            SMF049 - JES INTEGRITY REPORT
                                                   SYSTEM  3083

KEY: TNF-TERM NOT DEFINED IP-INVALID PASSWORD LASO-LINE ALREADY SIGNED ON TASO-TERM ALREADY SIGNED ON

                              INVALID
REMOTE           LINE        PASSWORD      DATE          TIME       REASON              MESSAGE

REMOTE*7        E*12          PASSWD34    89/04/26      13:51:37    TND                 MESSAGE*7
REMOTE TOTAL

REMOTE*4        E*4E          PASSWD96    89/04/26      13:51:38    IP                  MESSAGE*90
REMOTE*4        E*4E          PASSWD32    89/04/26      13:51:39    TND                 MESSAGE*7
REMOTE TOTAL

SYSID TOTAL

FINAL TOTAL
```

This sample lists information concerning JES integrity in the SMF data set.

VSAM Opens — SMF062

The SMF062 routine creates a report listing each VSAM data set opened and each VSAM open attempt that failed due to an invalid password. The report is sequenced by TIME, DATE, and CLUSTER (VSAM data set name) within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF062A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF062B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting VSAM data set open activity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on VSAM data set open activity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on VSAM data set open activity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on VSAM data set open activity. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF062.

Input

```
%SMFILE
%SMF062A 050000 080000 891219 891219
%SMF062B
```

Output

```
                                SMF062 - VSAM OPEN REPORT                         PAGE      1
                                          SYSTEM 3081

                     CLUSTER                    VOLUME                                     ERROR
                     CATALOG                    SERIAL  JOBNAME    DATE      TIME          FLAG

   CATALOG.VUSR031                              USR031  JACKSON  89/12/19  7:42:17
   CATALOG.VUSR031
   USERCAT.VUSR012                              USR012  JACKS02  89/12/19  7:35:34
   USERCAT.VUSR012
   VCMF1.JACKSON.CMF21A.DATABASE                USR031  JACKSON  89/12/19  7:42:20
   CATALOG.VUSR031
   VCMF1.JACKSON.CMF21A.JOURNAL                 USR031  JACKSON  89/12/19  7:42:19
   CATALOG.VUSR031
   VLIBSYS.APT.R012ABS.APTCTL                   USR012  JONES    89/12/19  7:51:30
   USERCAT.VUSR012
   VLIBSYS.APT.R012ABS.APTLIBC                  USR012  JONES    89/12/19  7:51:32
   USERCAT.VUSR012
   VSALTLAK.TRNPRIM                             USR025  CICS     89/12/19  5:09:04
   CATALOG.VUSR025

SYSID TOTAL

FINAL TOTAL
```

This example lists VSAM data set open activity in the SMF data set.

VSAM Deletes — SMF067

The SMF067 routine creates a report that lists deleted VSAM entries (components, clusters, paths, and so on). The report indicates the type of delete activity performed, for example, scratch, delete, and path delete. The report is sequenced by ENTRY within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF067A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF067B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on deleted VSAM entries. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on deleted VSAM entries. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on deleted VSAM entries. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on deleted VSAM entries. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF067.

Input

```
%SMFILE
%SMF067A 000000 130000 890319 890322
%SMF067B
```

Output

```
                                         SMF067 - DELETED VSAM ENTRIES REPORT
                                                      SYSTEM 3083

                   ENTRY                                                   FUNCTION
                  CATALOG                  JOBNAME     DATE      TIME      COMPLETED      STRUCTURE

SYS.PANLINK.BOX1                            JONESPL    89/03/20  11:58:01  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.PANLINK.BOX1                            JONESPL    89/03/20  12:07:39  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.PANLINK.BOX1.DATA                       JONESPL    89/03/20  12:07:39  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYS.PANLINK.BOX1.DATA                       JONESPL    89/03/20  11:58:01  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYS.PANLINK.BOX2                            JONESPL    89/03/20  12:03:31  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.PANLINK.BOX2                            JONESPL    89/03/20  12:12:20  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.PANLINK.BOX2.DATA                       JONESPL    89/03/20  12:12:20  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYS.PANLINK.BOX2.DATA                       JONESPL    89/03/20  12:03:31  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYS.PANLINK.TEST                            JONESPL    89/03/20  12:14:06  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.CATLIBU                                 JOHNSON    89/03/20  12:36:59  UNCATALOGED    VSAM CLUSTER
USERCAT.XUSER1B

SYS.CATLIBU.DATA                            JOHNSON    89/03/02  12:36:59  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYS.CATLIBU.INDEX                           JOHNSON    89/03/20  12:37:01  UNCATALOGED    VSAM INDEX
USERCAT.XUSER1B

VSAMDSET.TFC7D476.DFD85078.T98DAFEA.TFC7D476 JONESPL   89/03/20  12:14:06  UNCATALOGED    VSAM DATA
USERCAT.XUSER1B

SYSID TOTAL


FINAL TOTAL
```

This example lists VSAM entries deleted in the SMF data set.

VSAM Renames — SMF068

The SMF068 routine creates a report listing renamed VSAM data sets. The report indicates both the old and new data set name. The report is sequenced by old data set name within system ID (SYSID). The parameters that you pass allow for selection based on date and time.

Syntax

```
%SMF068A starttime endtime startdate enddate [REPORT reportname]
```

```
%SMF068B [REPORT reportname]
```

`starttime`

Specify the time that the routine will begin reporting on renamed VSAM data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`endtime`

Specify the ending time for reporting on renamed VSAM data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in HHMMSS format based on a 24-hour clock.

`startdate`

Specify the date that the routine will begin reporting on renamed VSAM data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`enddate`

Specify the ending date for reporting on renamed VSAM data sets. A valid value is an actual numeric value or the name of a field containing a numeric value. Express values in YYMMDD format.

`[REPORT reportname]`

For the second and all subsequent invocations of the same SMF routine, specify a unique reportname that associates the processing performed in the A routine with the report of the B routine. For each reportname specified on an A routine, the identical reportname must be specified on a B routine of the same SMF report type. For the first occurrence of an SMF routine, the routine supplies a default name and does not require the parameter.

Operation

To include the required file and field definitions, all SMF audit routines require you to invoke the SMFILE routine prior to the audit routine. If you invoke multiple SMF routines, you must invoke SMFILE only once, followed by the invocation of the desired SMF routines. For details, see SMF Audit Routines earlier in this chapter.

Example

The following is an example of SMF068.

Input

```
%SMFILE
%SMF068A 130000 140000 890426 890426
%SMF068B
```

Output

```
                                        SMF068 - RENAMED VSAM ENTRIES REPORT
                                                     SYSTEM 3083

                OLD DS NAME
                  CATALOG                           NEW DS NAME             JOBNAME     DATE       TIME

JONES.VSAM.FILE                     JONES.SAVE.FILE                          JONES     89/04/26   13:51:38
USERCAT.XUSER1B

SAMPLE.FILE                         SAMPLE.VSAM.FILE                         JOHNSON   89/04/26   13:51:39
USERCAT.XUSER1B

SMITH.TEST1                         SMITH.PROD                               SMITHVS   89/04/26   13:51:37
USERCAT.XUSER1B

SYSID TOTAL


FINAL TOTAL
```

This sample lists VSAM data sets renamed in the SMF data set.

Frequency Distribution of SMF Records — SMFCNT

The SMFCNT routine creates a frequency distribution of all SMF record types on the input SMF file. No selection criteria are available.

Syntax

```
%SMFCNT
```

Operation

You can use the SMFCNT routine before executing any of the other SMF audit routines. The SMFCNT routine provides a frequency distribution of all SMF record types. If there are no records of a specific type present on the SMF data set, then the report does not list a frequency distribution for that record type.

The SMFCNT routine contains the file definitions from SMFILE. Therefore, you must not invoke the SMFILE routine prior to SMFCNT. Also, you may not use SMFCNT with the other SMF reporting routines.

Example

The following is an example of SMFCNT.

Input

```
%SMFCNT
```

Output

```
                             FREQUENCY DISTRIBUTION OF TYPE                          PAGE     1
                                     INPUT FILENAME SMFILE

  TYPE    COUNT     PCT

    2         1      .0
    3         1      .0
    4     1,439     6.3      ******
    5       466     2.0      **
    6        82      .4
    9         4      .0
   10         3      .0
   11         7      .0
   14     5,695    25.0      *************************
   15     3,795    16.7      *****************
   17     2,070     9.1      *********
   18         4      .0
   19        64      .3
   20       505     2.2      **
   21       185      .8      *
   22         1      .0
   26       905     4.0      ****
   30     2,484    10.9      ***********
   32        36      .2
   34        37      .2
   35        37      .2
   40     2,206     9.7      **********
   50        25      .1
   57       271     1.2      *
   60       476     2.1      **
   61         3      .0
   62       682     3.0      ***
   63         3      .0
   64     1,266     5.6      ******
   65         5      .0
   66         3      .0
   90         1      .0
         22,762   100.0
```

This sample lists the frequency of occurrence of the various types of SMF records contained on the input SMF file.

SMF Record Field Definitions

The SMF record field definition macros provide field definitions for the majority of SMF record types. A macro has been created for most SMF record types which contain field definitions that follow the naming conventions listed in the IBM guide *OS/390 MVS System Management Facilities (SMF)*. With the release of MVS/XA 2.2.0 and MVS/ESA 3.1.3, some SMF records changed. To reflect these changes, three sets of CA-PanAudit Plus macros define the SMF records. The functionality of these macros is identical to their predecessors; they define the MVS/XA 2.2.0 or MVS/ESA 3.1.3 SMF record layouts.

You invoke these macros by coding a percentage sign (%), then the appropriate macro name. The following is a list of the macro names, the SMF record types, and a description of the SMF record types. The macros listed in column –A- are to be used with SMF records generated by MVS systems prior to MVS/XA 2.2.0 or MVS/ESA 3.1.3. Those listed in column –B- are for SMF records produced by MVS/XA 2.2.0. Those listed in column –C- are for SMF records produced by MVS/ESA 3.1.3.

Macro Name -A-	Macro Name -B-	Macro Name -C-	SMF Record Type	SMF Record Description
SMFILE	S22FILE	SE13FILE	-	SMF Data File Definitions for Audit Routines
SMFR00	S22R00	SE13R00	0	IPL
SMFR02	S22R02	SE13R02	2	Dump Header
SMFR03	S22R03	SE13R03	3	Dump Trailer
SMFR04	S22R04	SE13R04	4	Step Termination
SMFR05	S22R05	SE13R05	5	Job Termination
SMFR06J2	S22R06J2	SE13R06A	6	JES2 Output Writer
SMFR06J3	S22R06J3	SE13R06B	6	JES3 Output Writer
SMFR07	S22R07	SE13R07	7	Data Lost
SMFR08	S22R08	SE13R08	8	I/O Configuration
SMFR09	S22R09	SE13R09	9	VARY ONLINE
SMFR10	S22R10	SE13R10	10	Allocation Recovery
SMFR11	S22R11	SE13R11	11	VARY OFFLINE
SMFR14	S22R14	SE13R14	14	INPUT or RDBACK Data Set Activity
SMFR15	S22R15	SE13R15	15	OUTPUT, UPDAT, INOUT or OUTIN Data Set Activity
SMFR17	S22R17	SE13R17	17	Scratch Data Set Status
SMFR18	S22R18	SE13R18	18	Rename Data Set Status
SMFR19	S22R19	SE13R19	19	Direct Access Volume
SMFR20	S22R20	SE13R20	20	Job Initiation
SMFR21	S22R21	SE13R21	21	Error Statistics by Volume
SMFR22	S22R22	SE13R22	22	Configuration
SMFR23	S22R23	SE13R23	23	SMF Status Record
SMFR25	S22R25	SE13R25	25	JES3 Device Allocation
SMFR26J2	S22R26J2	SE13R26A	26	JES2 Job Purge
SMFR26J3	S22R26J3	SE13R26B	26	JES3 Job Purge
SMFR30	S22R30	SE13R30	30	Common Address Work Record
SMFR31	S22R31	SE13R31	31	TIOC Initialization
SMFR32	S22R32	SE13R32	32	TSO User Work Accounting Record
SMFR34	S22R34	SE13R34	34	TS-Step Termination
SMFR35	S22R35	SE13R35	35	LOGOFF
SMFR40	S22R40	SE13R40	40	Dynamic DD
SMFR43J2	S22R43J2	SE13R43A	43	JES2 Start
SMFR43J3	S22R43J3	SE13R43B	43	JES3 Start
SMFR45J2	S22R45J2	SE13R45A	45	JES2 Withdrawal
SMFR45J3	S22R45J3	SE13R45B	45	JES3 Stop

SMFR47J2	S22R47J2	SE13R47A	47	JES2 SIGNON/Start Line (BSC only)
SMFR47J3	S22R47J3	SE13R47B	47	JES3 SIGNON/Start Line LOGON
SMFR48J2	S22R48J2	SE13R48A	48	JES2 SIGNOFF/Stop Line (BSC only)
SMFR48J3	S22R48J3	SE13R48B	48	JES3 SIGNOFF/Stop Line LOGON
SMFR49J2	S22R49J2	SE13R49A	49	JES2 Integrity (BSC only)
SMFR49J3	S22R49J3	SE13R49B	49	JES3 Integrity
SMFR50	S22R50	SE13R50	50	ACF/VTAM Tuning Statistics
SMFR52	S22R52	SE13R52	52	JES2 LOGON/Start Line (SNA only)
SMFR53	S22R53	SE13R53	53	JES2 LOGOFF/Stop Line (SNA only)
SMFR54	S22R54	SE13R54	54	JES2 Integrity (SNA only)
SMFR55	S22R55	SE13R55	55	JES2 Network SIGNON Record
SMFR56	S22R56	SE13R56	56	JES2 Network Integrity Record
SMFR57J2	S22R57J2	SE13R57A	57	JES2 Network SYSOUT Transmission Record
SMFR57J3	S22R57J3	SE13R57B	57	JES3 Network SYSOUT Transmission Record
SMFR58	S22R58	SE13R58	58	JES2 Network SIGNOFF Record
SMFR62	S22R62	SE13R62	62	VSAM Component or Cluster Opened
SMFR63	S22R63	SE13R63	63	VSAM Entry Deleted
SMFR64	S22R64	SE13R64	64	VSAM Component or Cluster Status
SMFR67	S22R67	SE13R67	67	VSAM Entry Deleted
SMFR68	S22R68	SE13R68	68	VSAM Entry Renamed
SMFR69	S22R69	SE13R69	69	VSAM Data Space Defined, Extended or Deleted
SMFR70	S22R70	SE13R70	70	CPU Activity
SMFR71	S22R71	SE13R71	71	Paging Activity
SMFR72	S22R72	SE13R72	72	Workload Activity
SMFR73	S22R73	SE13R73	73	Channel Activity
SMFR74	S22R74	SE13R74	74	Device Activity
SMFR75	S22R75	SE13R75	75	Page/Swap Data Set Activity
SMFR76	S22R76	SE13R76	76	Trace Activity
SMFR77	S22R77	SE13R77	77	Enqueue Activity
SMFR79	S22R79	SE13R79	79	Monitor II Activity
SMFR82	S22R82	SE13R82	82	Security
SMFR90	S22R82	SE13R90	90	System Status Record

Operation

These macros contain the field definitions for the SMF record types. The use of these macros with CA-Easytrieve Plus logic eliminates the time-consuming task of defining the fields that the SMF records contain.

These macros do not contain a CA-Easytrieve Plus FILE statement. You must supply the FILE statement and all associated processing logic. The following example invokes SMF field definition macros:

```
FILE SMFFILE
%SMFR20
%SMFR05
%SMFR04
SMFTYPE   2  1  B
JOB INPUT SMFFILE
IF SMFTYPE NE 20, 4, 5
   GO TO JOB
END-IF
   .
   .
user-defined processing
   .
   .
```

In this example, the file SMFFILE contains the record definitions for Job Initiation (SMFR20), Job Termination (SMFR05), and Step Termination (SMFR04). A common field name, SMFTYPE, provides a facility to screen the input file to accept only record types 20, 5, and 4. The IF statement screens the input file by testing the SMFTYPE field. If the record type is not equal to 20, 5, or 4, the GO TO JOB statement transfers control to the JOB statement and effectively bypasses the user-defined processing for that record.

Fields defined in the SMF record field definition macros follow the exact layout as the IBM SMF Guide defines, with the following exceptions:

- Many SMF records contain variable-length sections. These sections usually contain a variable number of fixed-length subsections. A field within the record contains the number of subsections which comprise a variable-length section. To provide a method for accessing this data with CA-Easytrieve Plus, the SMF record field definition macro uses an indexing technique. See the CA-Easytrieve Plus *Reference Guide* for details regarding the use of indexing.
- Several fields in SMF records are defined with data structures not currently supported by CA-Easytrieve Plus . These include:
  - Variable-length fields
  - Fields greater than 254 bytes
  - Binary fields greater than four bytes in length

The SMF record field definition macro uses different methods to provide definitions for these nonsupported data types depending on the use of the information. For information regarding the methods used, see the IBM SMF Guide and the appropriate SMF record field definition macro.

Chapter 10

Advanced SMF Reporting Facility (JIF)

The OS Job Information Facility (JIF) is a system for reporting on records obtained from IBM System Management Facilities (SMF).

Processing SMF-generated data for use in statistical analysis, cost accounting, and customer billing may very easily become an application nightmare. JIF retrieves SMF records, consolidates them, creates files of SMF data, and produces reports on this data without the need for you to develop sophisticated application software to interface with SMF.

For most effective use of JIF, you must have knowledge of SMF records and know the SMF parameters in effect at your installation.

JIF Capabilities

JIF lets you:

- Report on SMF record types 00, 04, 05, 06, 07, 20, 26, and 40. Optionally, report on record types 34 and 35, or type 30.
- Report on other SMF record types through the user exit facility.
- Consolidate the SMF data into job and, optionally, TSO session representations.
- Create an SMF data file tailored to your needs.
- Produce preformatted statistical reports using the supplied routines.
- Create customized reporting routines of your own with CA-Easytrieve Plus .
- Receive audit reports on your use of the JIF system.
- Report on SMF records generated before MVS/XA 2.2.0, or SMF records generated by MVS/XA 2.2.0 or MVS/ESA 3.1.3 and above.

Facility Description

JIF has five components:

- JIFOPTS Options Module
- JIFSEL SMF Data Processor Function
- A User Exit Facility
- A CA-Easytrieve Plus Read Input Exit (JIFRDREX)
- Statistical Reporting Routines

JIFOPTS

The options module, JIFOPTS, provides information to JIFSEL indicating which of the SMF record types you want processed and the content of the consolidated record file to be produced. This provides a degree of customization in the consolidated file.

JIFSEL

JIFSEL processes the data produced by SMF, consolidating all SMF records for a job or TSO session into a single record. This record is then written to the consolidated file. JIFSEL selects both automatically and on the basis of the options you specify in JIFOPTS.

User Exit Facility

The user exit facility gives you the ability to further customize the consolidated file produced by JIFSEL. Use the EXIT1 routine to select and the EXIT2 routine to process any additional SMF record types you want to report on.

Read Input Exit

JIFRDREX reads the consolidated file, then formats and presents a fixed-length record to CA-PanAudit Plus. The JIF routines can be used to generate reports on this file.

Statistical Reporting Routines

CA-PanAudit Plus routines are provided, which allow you to produce a variety of statistical reports from the data in the consolidated file. You can create additional customized reporting routines using CA-Easytrieve Plus .

Facility Operation

Execution of JIF is a two-step process:

1. Execute JIFSEL, which creates the consolidated file.
2. Read the consolidated file and produce a report by invoking a CA-PanAudit Plus routine.

The following JCL illustrates this process. The first step executes JIFSEL to create the consolidated file and audit file from the SMF data. The second executes CA-Easytrieve Plus and invokes a CA-PanAudit Plus routine, OSJIF03.

```
//jobname   JOB  accounting.info
//STEP1     EXEC PGM=JIFSEL
//STEPLIB   DD   ...
//SYSPRINT  DD SYSOUT=A
//SYSUDUMP  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//PJPRINT   DD SYSOUT=A
//PJSORTIN  DD DSN=user.SMF.dataset,DISP=(OLD,KEEP),UNIT=TAPE,
//             VOL=SER=xxxxxx
//SORTOUT   DD DSN=JIF.consol.idated.file,DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(10,10),RLSE),UNIT=SYSDA,VOL=SER=xxxxxx
//PJAUDIT   DD DSN=JIF.audit.file,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(1,1),RLSE),UNIT=SYSDA,VOL=SER=xxxxxx
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,5)
/*
//STEP2     EXEC PGM=EZTPA00
//STEPLIB   DD ...
//PANDD     DD DSN=PAPL.macro.library,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,5)
//CONSOL    DD DSN=JIF.consol.idated.file,DISP=SHR
//AUDIT     DD DSN=JIF.audit.file,DISP=SHR
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,100))
//SYSIN     DD *
%JIFREC YYNNNNNN
%OSJIF03 84003 84104
/*
//
```

MVS/XA 2.2.0 Users

For users wanting to use MVS/XA 2.2.0 SMF records, append ,PARM='MVS(X220)' to the line that starts with //STEP1, and verify that the SMF record input file contains only records from that release of MVS SMF. The default is pre-MVS/XA 2.2.0 SMF records so that existing operational job streams will run without modification.

Example:

```
//STEP1      EXEC PGM=JIFSEL,PARM='MVS(X220)'
```

Pre-MVS/XA 2.2.0 SMF or MVS/XA 2.2.0 SMF and later routines can both be used on a processor running pre-2.2.0, 2.2.0, or post-2.2.0 releases of MVS as long as the appropriate data for the level of JIF/SMF routines selected is used. Both sets of routines can be run on the same processor, given that the previous parm is appended to the JCL.

MVS/ESA 3.1.3 Users

For users wanting to use MVS/ESA 3.1.3 SMF records, append ,PARM='MVS(E313)' to the line that starts with //STEP1, and verify that the SMF record input file contains only records from that release of MVS SMF. The default is pre-MVS/ESA 3.1.3 SMF records so that existing operational job streams will run without modification.

Example:

```
//STEP1      EXEC PGM=JIFSEL,PARM='MVS(E313)'
```

Pre-MVS/ESA 3.1.3 SMF or MVS/ESA 3.1.3 SMF and later routines can both be used on a processor running pre-3.1.3, 3.1.3, or post-3.1.3 releases of MVS as long as the appropriate data for the level of JIF/SMF routines selected is used. Both sets of routines can be run on the same processor, given that the previous parm is appended to the JCL.

JIFOPTS

Certain JIFSEL features are optional. The load module that specifies all options is JIFOPTS.

At installation, a model JIFOPTS module that contains all defaults is established. If your environment requires options other than those JIFOPTS supplies, you must link edit a new JIFOPTS.

For details on how to relink JIFOPTS, and for a description of the default and optional values, see the CA-PanAudit Plus *Installation Guide*.

JIFSEL

JIFSEL has three functional phases during its execution: record selection, sorting, and data consolidation.

SMF Record Selection — During the record selection phase, JIFSEL loads JIFOPTS to determine which SMF record types are to be selected. JIF processes certain SMF record types based on defaults. Others are selected by your exit routines.

Each time an SMF record is read, JIFSEL determines if the record is selected for processing. If selected, JIFSEL builds a sort key from the record. The record is then passed to your operating system sort. If not selected, the program defined by the EXIT1 parameter of the JIF options table is invoked.

Depending on the action taken by your exit, the SMF record is either bypassed from further processing or forced into the processing stream. When a record is forced, JIFSEL builds the sort key, and the record is passed on to your sort.

Sorting by Job or TSO Session — JIFSEL builds a 28-byte sort key from information in the SMF record. This is done for all records processed. The actual structure of the JIF sort key is discussed in the CA-PanAudit Plus *Installation Guide.* You must consider the structure of the sort key when forcing some types of SMF records. The SMF records selected by JIFSEL or by your exit are sorted into a chronological order by job or TSO session.

Consolidating SMF Records — After the sort, JIFSEL collects information from multiple SMF records into a single consolidated record. (See Consolidated Record Fields later in this chapter.)

During this phase of operation, JIF communicates with the program defined by the EXIT2 parameter in JIFOPTS To process records selected by your EXIT1 routine. If EXIT2 is not specified, JIFSEL bypasses any records that are forced by EXIT1 when writing the consolidated record.

SMF Record Selection

JIFSEL selects SMF records in three ways:

- By default
- Through parameters specified in the options macro
- By your exit routines

The record types selected by each of these techniques are shown in the following table.

Jifsel Default	Options Macro	User Exit Facility
00, 04, 05 06, 07, 20 26, 40	30 (subtypes 01, 04, 05) Batch and TSO; 34, 35 (TSO)	All other SMF record types

The left-hand column lists the SMF record types processed by JIFSEL.

Column two indicates additional record types you can process automatically by modifying parameters in the options module. For example,

- Type-30 records can be processed instead of types 20, 04, 40, and 05
- TSO data (types 34 and 35) can be processed

Column three indicates the SMF record types you can process by means of your exit routines.

Batch Environment

JIFSEL processes eight SMF record types from the batch environment:

```
Type 00 - IPL
Type 04 - Step Termination (batch job)
Type 05 - Job Termination (batch job)
Type 06 - JES2 or JES3 Output Writer
Type 07 - Lost Data
Type 20 - Job Initiation (batch job)
Type 26 - JES2 or JES3 Job Purge
Type 40 - Dynamic DD
```

Optionally, you can process type-30 (common address space work area) records. For details on the JIF options table, see the CA-PanAudit Plus *Installation Guide*.

TSO Environment

If the option TSO=YES is in effect, JIFSEL processes the following SMF record types in addition to those listed on the previous page:

```
Type 20 - TSO Job Initiation
Type 34 - TSO Step Termination
Type 35 - Logoff
```

Type-30 records can also be processed for TSO environments. Set the options macro parameter SMF30=YES and concurrently set TSO=YES.

SMF Record Content

The SMF data in the consolidated file, as a result of JIFSEL execution, is derived from the SMF record sources described in the following table.

Definitions for and descriptions of these records can be found in the appropriate IBM guides on System Management Facilities (SMF).

SMF Record Type	Record Name	Record Contents
20	Job Initiation	Job name System identification User identification Programmer's name Accounting information (first 24 positions)
04		Step termination Step name Program name Job name System identification Step start date and time Step termination date and time Step completion code Storage allocation Storage used Step CPU time (for MVS this represents SRB + TCB times) Step CPU time under SRB (MVS only) Step CPU time under TCB Device counts; EXCP counts for devices Pageins and pageouts Number of address space swap sequences Number of VIO pageins and pageouts Number of service units Residence times Number of page seconds

SMF Record Type	Record Name	Record Contents
05	Job Termination	Job name System identification Job termination time and date Job start time and date Number of card images read by job Job priority Resident time Job input class Storage protect key Job CPU time (for MVS this represents SRB + TCB times) Job CPU time under SRB (MVS only) Job CPU time under TCB Job transaction active time Performance group number of last step
06	Output Writer	Time and date output was completed System identification Job name Sysout class Writer start date and time Number of logical records written Form number Approximate page count Logical device name
07	Data Lost	Number of type-07 records processed during driver execution. For each 07 record processed: ■ Date and time record loss began ■ Date and time records stopped ■ Number of records lost
40	Dynamic D.D.	Information is equivalent to the dataset parts of Type-04 records (device counts and EXCP counts for the devices).

SMF Record Type	Record Name	Record Contents
30	Common Address Space Work Area	Information is the same as the record types (20, 04, 05, and 40) it replaces. SMF type-30, subtypes 1, 4, 5 are processed in place of SMF types 20, 04, 05, and 40.
34	TSO Step termination	Essentially the same information as type-04 records. Two additional fields are the number of lines of terminal output (number of TPUTS issued) and the number of lines of terminal input (number of TGETS satisfied).
35	Logoff	Essentially the same information as type05 records. Two additional fields are the number of lines of terminal output (number of TPUTS issued) and the number of lines of terminal input (number of TGETS satisfied).
26	Job Purge	System identification Job name Job number Job class Job priority Number of input cards for job

JIFSEL Data Sets

JIFSEL creates three output data sets: the consolidated record file, the audit file, and the audit report.

Consolidated Record File—The JIF consolidated file contains one record for each job. If requested, each record can contain individual step and spool information.

Each record contains a 256-position user portion for data you include through exit processing.

Audit File—A record is created for the audit file each time JIFSEL is executed. The record contains input data set name, number of records read, Initial Program Load (IPL) records (SMF type-00) read, data lost (SMF type-07 records), and number of records processed.

Audit Report—The JIF audit report is a report on the data recorded and stored in the JIF audit file.

Consolidated Record Fields

The following pages provide you with the field names used by the JIF routines when accessing the consolidated file. The file and field definitions are contained in the macro JIFREC. When using these fields, See the parameters of the JIFOPTS options module, as the contents of certain fields are developed based on parameters specified in JIFOPTS.

The description that follows describes the record presented to the JIF routines.

The area beginning at position 642 in the record, labeled Scratch Pad Area, is used by the routines to store additional information in the records prior to sorting.

```
FILE CONSOL   EXIT (JIFRDREX  USING ('&FLAGS'))  WORKAREA (1024)
**********************************************************
*                                                        *
*         COMPUTER ASSOCIATES INCORPORATED               *
*                                                        *
*     OS JOB INFORMATION FACILITY VERSION 1.0            *
*                                                        *
*   THE FOLLOWING MACRO DESCRIBES THE INPUT RECORD       *
*   AS SEEN BY EASYTRIEVE. THE FIRST EIGHT BYTES         *
*   ARE AVAILABILITY FLAGS INDICATING WHEN PORTIONS      *
*   OF THE RECORD ARE PRESENT.                           *
*   XFLAG1 = Y WHEN JOB PORTION PRESENT.                 *
*   XFLAG2 = Y WHEN STEP PORTION PRESENT.                *
*   XFLAG3 = Y WHEN SPOOL PORTION PRESENT.               *
*   XFLAG4 THROUGH XFLAG7 RESERVED.                      *
*   XFLAG8 = Y WHEN USER PORTION PRESENT.                *
*                                                        *
**********************************************************
XFLAG1     1  1 A . XFLAG2     2  1 A . XFLAG3     3  1 A .
XFLAG4     4  1 A . XFLAG5     5  1 A . XFLAG6     6  1 A .
XFLAG7     7  1 A . XFLAG8     8  1 A .
***********************************************************
*                                                         *
*   THIS SECTION OF THE MACRO DESCRIBES THE JOB PORTION.  *
*                                                         *
***********************************************************
CMSOURCE      9       1   A
CMFLAG1      10       1   A
CMFLAG2      11       1   A
CMFLAG3      12       1   A
CMFLAG4      13       1   B
CMFLAG5      14       1   B
CMFLAG6      15       1   B
CMFLAG7      16       1   B
CMTOTSTP     17       2   B  0
CMOFFSTP     19       2   B
CMTOTSPL     21       2   B  0
CMOFFSPL     23       2   B
CMCOUNT1     25       4   B
CMCOUNT2     29       4   P
CMSYSID      33       4   A
CMJOBNM      37       8   A       HEADING ('JOB' 'NAME')
CMJOBNO      45       2   U       HEADING ('JOB' 'NO.')
CMJOBPT      47       2   U       HEADING ('P' 'T' 'Y')
CMJOBCL      49       1   A       HEADING ('C' 'L')
CMPRKEY      50       1   U
CMPRGNAM     51      20   A       HEADING ('PROGRAMMER' 'NAME')
CMPRGNSH     51      14   A
CMUSRID      71       8   A       HEADING ('USER' 'INFORMATION')
```

```
CMACCT       79      24   A
CMABEND     103       1   A
CMDTSTT     104       3   U       MASK ('99/99/99')
CMTMSTT     107       3   U       MASK ('Z9:99:99')  +
                                  HEADING ('START' 'TIME')
CMTMSTP     110       3   U       MASK ('Z9:99:99')  +
                                  HEADING ('STOP' 'TIME')
CMTMELAP    113       5   U  4    HEADING ('ELAPSED' 'TIME')
CMTMCPU     118       4   U  4    HEADING ('CPU' 'TIME')
CMTMTCB     122       4   U  4    HEADING ('TCB' 'TIME')
CMTMSRB     126       4   U  4    HEADING ('SRB' 'TIME')
CMTMACT     130       5   U  4    HEADING ('ACTIVE' 'TIME')
CMTMRES     135       5   U  4    HEADING ('RESIDENT' 'TIME')
CMTMALD     140       2   U  2    HEADING ('ALLOCATION' 'TIME')
CMTMRDR     142       4   U  4    HEADING ('RDR' 'TIME')
CMTMWRQ     146       4   U  4    HEADING ('WRITER' 'TIME')
CMTMWTR     150       4   U  4
CMTMTURN    154       5   U  4    HEADING ('TURNAROUND' 'TIME')
CMCARDR     159       3   U  0
CMCARDP     162       3   U  0
CMLINES     165       4   U  0    HEADING ('LINES' 'PRINTED')
CMTAPE      169       1   U  0
CMDISK1     170       1   U  0
CMDISK2     171       1   U  0
CMDISKX     172       1   U  0
CMOTHER     173       1   U  0
CMIOTP      174       3   U  0    HEADING ('TAPE I/O' 'COUNT')
CMIODK1     177       3   U  0
CMIODK2     180       3   U  0
CMIODKX     183       3   U  0
CMIOOTH     186       3   U  0
CMVIOIN     189       3   U  0
CMVIOOT     192       3   U  0
CMSWAPIN    195       3   U  0
CMSWAPOT    198       3   U  0
CMPAGEIN    201       3   U  0    HEADING ('PAGEIN' 'COUNT')
CMPAGEOT    204       3   U  0    HEADING ('PAGEOUT' 'COUNT')
CMADDSP     207       2   U  0    HEADING ('SWAP' 'COUNT')
CMPGSEC     209       4   U  0    HEADING ('PAGE' 'SECONDS')
CMSERV      213       4   U  0    HEADING ('SERVICE' 'UNITS')
CMPERFGP    217       2   U       HEADING ('PER.' 'GROUP')
CMCOREAL    219       4   B  0
CMCOREUS    223       2   B  0
CMTSOTG     225       4   U  0    HEADING ('TSO' 'GETS')
CMTSOTP     229       4   U  0    HEADING ('TSO' 'PUTS')
DATE        104       1   U
STRTTM      107       1   U
CNTTAPE     169       1   A
**********************************************************
*                                                        *
*   THIS SECTION DESCRIBES THE JOB STEP PORTION.         *
*                                                        *
**********************************************************
CSNAME      233       8   A       HEADING ('STEP' 'NAME')
CSPROGN     241       8   A       HEADING ('PROGRAM' 'NAME')
CSCOREAL    249       4   B  0
CSCOREUS    253       2   B  0
CSABND      255       4   A       HEADING ('ABEND' 'CODE')
CSSTDT      259       3   U       MASK ('99/99/99')  +
                                  HEADING ('STEP' 'RUN' 'DATE')
CSSTTM      262       3   U       MASK ('99:99:99')  +
                                  HEADING ('STEP' 'START' 'TIME')
CSSPTM      265       3   U       MASK ('99:99:99')
CSELAPT     268       4   U  4    HEADING ('STEP' 'ELAPSED' 'TIME')
CSCPU       272       3   U  4    HEADING ('STEP' 'CPU' 'TIME')
CSTCB       275       3   U  4
```

```
CSSRB       278       3   U  4
CSACT       281       4   U  4
CSRES       285       4   U  4
CSALDEL     289       2   U  2
CSTAPNUM    291       1   U  0
CSDISK1     292       1   U  0
CSDISK2     293       1   U  0
CSDISKX     294       1   U  0
CSOTHNM     295       1   U  0
CSIOTP      296       3   U  0
CSIODK1     299       3   U  0
CSIODK2     302       3   U  0
CSIODKX     305       3   U  0
CSIOOTH     308       3   U  0
CSVIOIN     311       3   U  0
CSVIOOT     314       3   U  0
CSSWAPIN    315       3   U  0
CSSWAPOT    320       3   U  0
CSPAGEIN    323       3   U  0
CSPAGEOT    326       3   U  0
CSADDSP     329       2   U  0
CSPGSEC     331       4   U  0
CSSERV      335       4   U  0
CSTSOTG     339       4   U  0
CSTSOTP     343       4   U  0
***********************************************************
*                                                         *
*   THIS SECTION DESCRIBES THE JOB OUTPUT SPOOL PORTION.  *
*                                                         *
***********************************************************
CPDURTN     347       4   U  4
CPLOGUN     351       3   U  0
CPPAGES     354       3   U  0
CPDEVNM     357       8   A
CPROUTE     365       2   B       MASK (HEX)
CPFORM      367       4   A
***********************************************************
*                                                         *
*   THIS SECTION DESCRIBES THE USER PORTION.              *
*                                                         *
***********************************************************
USERAREA    371     254   A
***********************************************************
*                                                         *
*   THIS SECTION DESCRIBES THE SCRATCH PAD PORTION.       *
*                                                         *
***********************************************************
*
CXPERLIN    646       4   U  2
CXPERCRD    650       4   U  2
CXURCST     654       5   U  2
*
CXCPUSRB    659       4   U  4
CXCPUTCB    663       4   U  4
CXCPUCST    667       6   U  2
*
CXIOTAPE    673       4   U  2
CXIODSK1    677       4   U  2
CXIODSK2    681       4   U  2
CXIODSKX    685       4   U  2
CXIOOTH     689       4   U  2
CXIODSKT    772       5   U  2
CXIOCST     773       5   U  2
*
CXCORUSE    703       2   B  0
CXCORALL    705       2   B  0
```

```
CXCORTOT    707       2   B  0
*
CXUNTAPE    709       3   U  2
CXUNDSK1    712       3   U  2
CXUNDSK2    715       3   U  2
CXUNDSKX    718       3   U  2
CXUNOTH     721       3   U  2
CXUNDISK    724       4   U  2
CXUNCST     728       4   U  2
*
CXWGTPTY    732       2   U  2
CXWGTCPU    734       2   U  2
*
CXPERTGET   736       3   U  2
CXPERTPUT   739       3   U  2
CXTPTGCST   742       4   U  2
CXCONCHG    746       4   U  2
*
FCLASS      754      12   A       HEADING ('JOB' 'CLASS')
TITLE1      754      10   A       HEADING ('TIME' 'INTERVAL')
TITLE2      754      14   A       HEADING ('CPU TIME' 'INTERVAL')
SERVUNIT    754      20   A       HEADING ('SERVICE' 'UNITS')
DEPTKEY      37       4   A
DEPTRES     804      31   A
DEPTFLD     804      20   A
DERROR      804      16   A
ERRDEPT     820       4   A
CREDKEY      37       2   A
CREDRES     844      33   A
CREDFLD     844       9   N  2
BUDGFLD     854       9   N  2
MONTKEY     926       2   N  0
MONTFLD     884      10   A       HEADING ('MONTH')
DEBIT       940       5   U  2
OVRDATE     106       1   U
DATE1       930       5   N
DATE2       935       5   N
OUTPG      1004       4   N       HEADING ('PER.' 'GROUP')
JIF11SRT   1008       1   A
PRTZERO    1009       2   P       MASK ('ZZ9')  +
                                  HEADING ('NUMBER' 'TAPES' 'ALLOCATED')
CXTOTIO    1011       4   U  2
***********************************************************
*                                                         *
*   THIS SECTION DESCRIBES THE WORKING STORAGE FIELDS     *
*   USED BY THE SAMPLE REPORTS SUPPLIED WITH THIS         *
*   FACILITY.                                             *
***********************************************************
CNT01         W       5   N  0    HEADING ('NUM.' 'OF' 'JOBS')
CNT02         W       5   N  0    HEADING ('NUM.' 'OF' 'SES.')
JOBCNT01      S       5   P  0    HEADING ('NUM.' 'OF' 'JOBS')
JOBCNT02      S       5   P  0    HEADING ('NUM.' 'OF' 'SES.')
JOBCNT03      S       5   P  0
JOBCNT04      S       5   P  0
JOBCNT06      S       5   P  0
JOBCOUNT      S       5   P  0
TOTCPU05      S       7   P  4
TOTCPU        S       7   P  4
DSKTOT08      W       5   P  0    HEADING ('DISK I/O' 'COUNT')
WORKDATE      W       8   A
DISKTOT       W       5   P  0    HEADING ('TOTAL' 'I/O' 'COUNT')
IOTOT         W       5   P  0    HEADING ('TOTAL' 'I/O' 'COUNT')
SWPCNT        W       5   P  0    HEADING ('TOTAL' 'PAGEIN' 'PAGEOUT')
PAGES         W       5   P  0    HEADING ('TOTAL' 'PAGES')
JPERCENT      W       6   N  3    HEADING ('PERCENT' 'EXECS')
SORTFLD       W      10   A
```

```
STJOB         W       8   A
DEBTOT        S       5   U  2
TOTACT        S       8   P  4    HEADING ('GRAND')
AVGCPU        W       4   P  4    HEADING ('AVG' 'CPU' 'TIME')
AVGACT        W       5   P  4    HEADING ('AVG' 'ACTIVE' 'TIME')
AVGRES        W       5   U  4    HEADING ('AVG' 'RESIDENT' 'TIME')
AVGELAP       W       5   P  4    HEADING ('AVG' 'CONNECT' 'TIME')
PCTACT        W       3   P  3    HEADING ('PCT' 'OF' 'TOTAL')
JOBNO         W       2   U  0    HEADING ('TSU' 'NO.')
HOLDKEY       S       4   A
HOLDKEY2      HOLDKEY 2   A
HOLDDESC      S       33  A
HOLDCRED      HOLDDESC     9  N  2
HOLDBUDG      HOLDDESC +10 9  N  2
DBTCHG        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
CRTAMT        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
TOTCHG        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
BGTAMT        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
BGTDEV        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
BGTPCT        W       6   P  2    MASK ('ZZZ,ZZZ,ZZZ.99-')
PRT_TALLY_    S       6   N  0    MASK ('ZZZZZ9-')
WORK_2N_      S       2   N  0
B_FLAG_       S       1   A       VALUE ('B')
C_FLAG_       S       1   A       VALUE ('C')
T_FLAG_       S       1   A       VALUE ('T')
Y_FLAG_       S       1   A       VALUE ('Y')
ZERO_         S       4   P  0    VALUE (0)
ONE_          S       4   P  0    VALUE (1)
TWO_          S       4   P  0    VALUE (2)
REPORT_FLAG_  S       3   A       VALUE ('NO ')
YES_          S       3   A       VALUE ('YES')
NO_           S       3   A       VALUE ('NO ')
N100_         S       4   P  0    VALUE (100)
N1000_        S       4   P  0    VALUE (1000)
N1999_        S       4   P  0    VALUE (1999)
N2000_        S       4   P  0    VALUE (2000)
N2999_        S       4   P  0    VALUE (2999)
N3000_        S       4   P  0    VALUE (3000)
N3999_        S       4   P  0    VALUE (3999)
N4000_        S       4   P  0    VALUE (4000)
N4999_        S       4   P  0    VALUE (4999)
N5000_        S       4   P  0    VALUE (5000)
N9999_        S       4   P  0    VALUE (9999)
N10000_       S       4   P  0    VALUE (10000)
N19999_       S       4   P  0    VALUE (19999)
N20000_       S       4   P  0    VALUE (20000)
N29999_       S       4   P  0    VALUE (29999)
N30000_       S       4   P  0    VALUE (30000)
N39999_       S       4   P  0    VALUE (39999)
N40000_       S       4   P  0    VALUE (40000)
N49999_       S       4   P  0    VALUE (49999)
N50000_       S       4   P  0    VALUE (50000)
N74999_       S       4   P  0    VALUE (74999)
N75000_       S       4   P  0    VALUE (75000)
N99999_       S       4   P  0    VALUE (99999)
N100000_      S       4   P  0    VALUE (100000)
```

Consolidated Record Field Descriptions

The following table describes each field name defined in the JIFREC macro:

Field Name	Description
XFLAG1	Indicates presence of main job section “Y” = job section present
XFLAG2	Indicates presence of step portion “Y” = step section present
XFLAG3	Indicates presence of spool section “Y” = spool section present
XFLAG4	Reserved
XFLAG5	Reserved
XFLAG6	Reserved
XFLAG7	Reserved
XFLAG8	Indicates presence of user section “Y” = user section present
COMSOURCE	Source of this job information “S” = SMF

Field Name	Description
CMFLAG1	State of this record: C = This record is complete, i.e., all SMF records required to create this record have been processed. B = This record is complete but was modified by the user. All SMF records required to create this record were processed, but the user in some way modified the record prior to its being written to the consolidated file. **Note:** Modification does not include addition of the user portion to the record. ●= This record is an orphan, i.e., all SMF records required to create this record have not been processed. X = This record is an orphan and was modified by the user. All SMF records required to create this record have not been processed, and the user in some way modified the record before it was written to the consolidated file. **Note:** Modification does not include addition of the user portion to the record.
CMFLAG2	Information source for this record: T = This record reports on TSO information. B = This record reports on batch job information. R = This record reports on batch information and is the product of a rerun situation.
CMFLAG3	User segment present indicator: Blank = The user segment is not present in this record. U = The user segment is present in this record.
CMFLAG4	Reserved
CMFLAG5	Reserved
CMFLAG6	Reserved
CMFLAG7	Reserved
CMTOTSTP	Number of steps in this job
CMOFFSTP	Reserved
CMTOTSPL	Number of spool records associated with this job
CMOFFSPL	Reserved
CMCOUNT1	Reserved
CMCOUNT2	Job start date in packed, Julian format (YYDDD)
CMSYSID	Job identification

Field Name	Description
CMJOBNM	Job name
CMJOBNO	Job number
CMJOBPT	Job priority
CMJOBCL	Job class
CMPRKEY	Protect key of job
CMPRGNAM	Programmer's name field from job card
CMUSRID	User identification
CMACCT	Accounting information from job card
CMABEND	Job abend indicator "Y" = a job step abend
CMDTSTT	Job start date
CMTMSTT	Job start time
CMTMSTP	Job stop time
CMTMELAP	Job elapsed time, in minutes to four decimal places
CMTMCPU	Job CPU time, in minutes to four decimal places
CMTMTCB	Job CPU time under a Time Control Block (TCB), in minutes to four decimal places
CMTMSRB	Job CPU time under a Service request block (SRB), in minutes to four decimal places
CMTMACT	Active time, in minutes to four decimal places
CMTMRES	Resident time, in minutes to four decimal places
CMTMALD	Allocation delay times, in seconds to two decimal places
CMTMRDR	Time job was on input queue, in minutes to four decimal places
CMTMWRQ	Time job was on output queue, in minutes to four decimal places
CMTMWTR	Writer duration time, in minutes to four decimal places
CMTMTURN	Turnaround time, in minutes to four decimal places
CMCARDR	Number of cards read by job
CMCARDP	Number of cards punched by job
CMILNES	Number of lines printed by job
CMTAPE	Number of tapes used by job

Field Name	Description
CMDISK1 MDISK2 CMDISKX	Number of disks used by job (in categories defined in JIFOPTS)
CMOTHER	Number of nondisk or nontape devices used by job
CMIOTP	EXCP count for tapes
CMIODK1 CMIODK2 CMIODKX	EXCP count disk devices (in categories defined in JIFOPTS)
CMIOOTH	EXCP count for other devices
CMVIOIN	I/O count for VIO-ins
CMVIOOT	I/O count for VIO-outs
CMSWAPIN	I/O count for swap-ins
CMSWAPOT	I/O count for swap-outs
CMPAGEIN	I/O count for page-ins
CMPAGEOT	I/O count for page-outs
CMADDSP	Number of address space swap sequences
CMPGSEC	Number of page seconds
CMSERV	Number of service units
CMPERFGP	Performance group number
CMCOREAL	Amount of core allocated for job, in 1KB units
CMCOREUS	Amount of core used by job, in 1KB units
CMTSOTG	The number of lines of terminal input (number of TGETS satisfied)
CMTSOTP	Number of lines of terminal output (number of TPUTS issued)
DATE	Overlay of start date field (CMDTSTT) to extract month
STRTTM	Overlay of start time field (CMTMSTT) to extract hour
CNTTAPE	Overlay of number of tape fields (CMTAPE) for a standard report
CSNAME	Step name
CSPROGN	Name of program executed in this (CSNAME) step
CSCOREAL	Amount of core allocated for step, in 1KB units
CSCOREUS	Amount of core used in step, in 1KB units

Field Name	Description
CSABND	Abend code if this step abended (Sxxx = system abend; otherwise, user code)
CSSTDT	Step start date
CSSTTM	Step start time
CSSPTM	Step stop time
CSELAPT	Step elapsed time, in minutes to four decimal places
CSCPU	CPU time for step, in minutes to four decimal places
CSTCB	CPU time for step under TCB, in minutes to four decimal places
CSSRB	CPU time for step under SRB, in minutes to four decimal places
CSACT	Active time for step, in minutes to four decimal places
CSRES	Resident time for step, in minutes, to four decimal places
CSALDEL	Allocation delay time, in seconds to two decimal places
CSTAPNUM	Number of tapes used in step
CSDISK1 CSDISK2 CSDISKX	Number of disks used by step according to categories defined in the options macro JIFOPTS
CSOTHNM	Number of nondisk or tape devices used in step
CSIOTOP	EXCP count for tapes in CSOTHNM
CSIODK1 CSIODK2 CSIODKX	EXCP count for disk devices used in CSOTHNM, in categories defined in options macro JIFOPTS
CSIOOTH	EXCP count for nondisk or tape devices used in this step
CSVIOIN	I/O count for VIO-ins
CSVIOOT	I/O count for VIO-outs
CSSWAPIN	I/O count for swap-ins
CSSWAPOT	I/O count for swap-outs
CSPAGEIN	I/O count for page-ins
CSPAGEOT	I/O count for page-outs
CSADDSP	Number of address space swap sequences
CSPGSEC	Number of page seconds

Field Name	Description
CSSERV	Number of service units
CSTSOTG	Number of lines of terminal input (number of TSGETS satisfied)
CSTSOTP	Number of lines of terminal output (number of TPUTS issued)
CPDURTN	Writer duration time, in minutes to four decimal places
CPLOGUN	Number of logical records written
CPPAGES	Number of pages of output produced
CPDEVNM	Device name
CPROUTE	Route codes
CPFORM	Forms identification
USERAREA	Defines the user section as one field SCRATCH PAD AREA
CXURCST	Result field of unit record cost calculation
CXCPUSRB	Cost of SRB CPU time for this record
CXCPUTCB	Cost of TCB CPU time for this record
CXIOTAPE	Cost of tape I/O
CXIODSK1 CXIODSK2 CXIODSKX	Cost of disk I/O categorized according to definitions in options macro JIFOPTS
CXIOOTH	Cost of other devices I/O activity
CXIODSKT	Total cost of disk I/O
CXIOCST	Total cost of all I/O applicable to this record
CXCORUSE	Cost of core used
CXCORALL	Cost of core allocated
CXCORTOT	Total cost related to core usage
CXUNTAPE	Cost related to tape unit allocation
CXUNDSK1 CXUNDSK2 CXUNDSKX	Cost related to disk unit. Usages by categories defined in the options macro JIFOPTS
CXUNOTH	Cost related to unit usage not defined previously
CXUNDISK	Total cost related to disk unit usage
CXUNCST	Total cost of all unit usage
CXWGTPTY	Cost weighted because of certain job priority

Field Name	Description
CXWGTCPU FCLASS	Cost weighted because of certain CPT usage
TITLE1 TITLE2 SERVUNIT	Used in certain reports as CONTROL fields
DEPTKEY	Key field for department tape lookup
DEPTRES	Result field for department table lookup
DEPTFLD	Overlay of result field for report printing
CREDKEY	Key field for credit table lookup
CREDRES	Result field for department table lookup
CREDFLD	Overlay of result field
BUDGFLD	Overlay of result field
MONTKEY	Key field for month table lookup
MONTFLD	Result field for month tape lookup
OVRDATE	Redefine date field to extract year
DEBIT	Work field in debit calculations
DATE1 DATE2	Work fields for picking up date parameters

Audit File

One audit file record is created each time JIFSEL is executed. The record contains four segments:

- A static portion where various count information is stored
- A data set section
- One IPL section for each SMF type-00 record processed
- One lost data section for each SMF type-07 record processed

A report on the contents of the record is produced by JIFSEL after its processing is completed.

The layout of the audit file record is shown in Audit Record Fields.

Audit Record Fields

The following lists the field definitions of the audit file. Descriptions of the field names are shown in the table that follows. An asterisk indicates breaks between the four record segments described on the previous page.

```
FILE AUDIT
AUDATE           1        3     U     MASK ('99/99/99')
AUTIME           4        3     U     MASK ('99:99:99')
AUSMFDTF         7        3     U     MASK ('99/99/99')
AUSMFTMF        10        3     U     MASK ('99:99:99')
AUSMFDTL        13        3     U     MASK ('99/99/99')
AUSMFTML        16        3     U     MASK ('99:99:99')
AUCMDTF         19        3     U     MASK ('99/99/99')
AUCMTMF         22        3     U     MASK ('99:99:99')
AUCMDTL         25        3     U     MASK ('99/99/99')
AUCMTML         28        3     U     MASK ('99:99:99')
AUSMREAD        31        5     P     MASK ('ZZZZZZZZ9')
AUSMRJCT        36        5     P     MASK ('ZZZZZZZZ9')
AUSMFRCD        41        5     P     MASK ('ZZZZZZZZ9')
AUCMCREA        46        4     P     MASK ('ZZZZZZ9')
AUCMDEL         50        4     P     MASK ('ZZZZZZ9')
AUCMMOD         54        4     P     MASK ('ZZZZZZ9')
AUCMORPH        58        4     P     MASK ('ZZZZZZ9')
AUCMRRUN        62        4     P     MASK ('ZZZZZZ9')
AUSMDUP         66        4     P     MASK ('ZZZZZZ9')
AUSDUMY         70        1     A
AUOFFSM0        71        2     B
AUOFFSM7        73        2     B
AUDSTOT         75        2     B
AUSM0TOT        77        2     B     MASK ('ZZZZ9')
AUSM7TOT        79        2     B     MASK ('ZZZZ9')
*  VS_ OCCURS FOR THE LENGTH OF THE LONGEST SINGLE SEGMENT
VS_             81        1     A     OCCURS  50
VS1_           VS_       50     A     INDEX SUB1
AUSDSNAM       VS1_      44     A
AUDSVOL        VS1_ +44   6     A
*
VS2_           VS_       12     A     INDEX  (SUB1, SUB2)
AUSM0SID       VS2_       4     A
AUIPLDT        VS2_ +04   3     U     MASK ('99/99/99')
AUIPLTM        VS2_ +07   3     U     MASK ('99:99:99')
AUSM0PT        VS2_ +10   1     B
AUSM0XX        VS2_ +11   1     A
*
VS3_           VS_       18     A     INDEX  (SUB1, SUB2, SUB3)
AUSM7SID       VS3_       4     A
AUTLOST        VS3_ +04   2     B
AULSTDT        VS3_ +06   3     U     MASK ('99/99/99')
AULSTTM        VS3_ +09   3     U     MASK ('99:99:99')
AULSFDT        VS3_ +12   3     U     MASK ('99/99/99')
AULSFTM        VS3_ +15   3     U     MASK ('99:99:99')
```

Audit Record Field Descriptions

The following table describes each field name in the audit record:

Name	Description
Static Section	
AUDATE	Date of run
AUTIME	Time of run
AUSMFDTF	Date of first SMF input record
AUSMFTMF	Time of first SMF input record
AUSMFDTL	Date of last SMF input record
AUSMFTML	Time of last SMF input record
AUCMDTF	Date of first consolidated record output
AUCMTFMF	Time of first consolidated record output
AUCMDTL	Date of last consolidated record output
AUCMTML	Time of last consolidated record output
AUSMREAD	Number of SMF records read
AUSMRJCT	Number of SMF records rejected
AUSMFRCD	Number of SMF records forced at EXIT1
AUCMCREA	Number of consolidated records created
AUCMDEL	Number of consolidated records deleted by user exit
AUCMMOD	Number of consolidated records modified by user exit
AUCMORPH	Number of consolidated records that are orphan records
AUCMRRUN	Number of rerun condition records
AUSMDUP	Number of duplicate data records
AUSDUMY	Reserved
AUOFFSMO	Offset of IPL section from start of record
AUOFFSM7	Offset of data lost section from start of record
AUDSTOT	Number of data set entries in record
AUSM0TOT	Number of IPL entries in record
AUSM7TOT	Number of data lost entries in record

Name	Description
Data Set Selection	
AUDSNAM	Data set name of input file to PANJOB
AUDSVOL	Volume serial number of AUDSNAM data set
IPL Section	
AUSMOSID	System identification of CPU experiencing IPL
AUIPLDT	Date of IPL
AUIPLTM	Time of IPL
AUSMOPT	SMF options in effect during execution of driver program
AUSM7SID	System identification of CPU from which data was lost
AUTLOST	Number of lost SMF records
AULSTDT	Starting date for lost records
AULSTTM	Starting time for lost records
AULSFDT	Finishing date for lost records
AULSFTM	Finishing time for lost records

Audit Report

The audit report is based on the audit record and is produced for each execution of JIFSEL. The report contains record counts, first and last dates and times of the SMF records processed, IPL information, and lost data information.

The following shows an example of the audit report produced by JIFSEL:

```
        03/22/84       COMPUTER ASSOCIATES - OS JOB INFORMATION FACILITY

PART ONE  ** INPUT  **

 -A-    DATASET PROCESSING      DATASET NAME   MONDAY.SMFRECS.DATA
                                VOLUME SERIAL  WORK01

 -B-    DATE/TIME               FIRST SMF RECORD     03/19/84     08:08:23
                                LAST  SMF RECORD     03/19/84     22:13:28

 -C-    RECORD TOTALS           SMF RECORDS  READ           28226
                                SMF RECORDS  REJECTED       22211
                                SMF RECORDS  FORCED             0
                                SMF RECORDS  DUPLICATE         23

 -D-    # I.P.L. RECORDS             1      DATE         TIME
                                          03/19/84     10:14:24

 -E-    # DATA LOST RECORDS          0
```

```
PART TWO  ** OUTPUT **

 -A-    RECORD TOTALS          CONSOLIDATED RECORDS  CREATED       881
                                                     DELETED         0
                                                     ORPHAN        174
                                                     RERUN           0
                                                     MODIFIED        0

 -B-    DATE/TIME     FIRST CONSOLIDATED RECORD    03/16/84     20:23:41
                      LAST  CONSOLIDATED RECORD    03/20/84     07:34:21
```

User Exit Facility

The user exit facility allows you to select and process all SMF records not automatically processed by JIFSEL. There are two entry points in the user exit facility. Each has a specific function:

EXIT1 — Allows you to code your own routines to select any SMF records not processed automatically. The EXIT1 facility is an extension of JIFSEL's record selection process.

EXIT2 — Is used to process the records selected by your EXIT1 routine. This consists of extracting data selected by the EXIT1 routine and inserting it into the User Area section provided in the consolidated record. The EXIT2 facility is an extension of JIFSEL's record consolidation function. This allows you to customize the record written to the consolidated file.

EXIT1

The supplied EXIT1 default (JIFEXIT1) has no effect on the SMF record selection performed by the driver program. JIFEXIT1 is a one-instruction program and simply returns to JIFSEL each time it is called.

To process SMF record types not provided in JIFSEL or the JIF options table, you must write a program and substitute it for the default. See the CA-PanAudit Plus *Installation Guide* for details on how to write your EXIT1 routine.

EXIT2

The supplied EXIT2 default routine (JIFEXIT2) has no effect on the contents of the consolidated file or on any of the functions performed by JIFSEL. JIFEXIT2 is a one-instruction program and simply returns to JIFSEL each time it is called.

Four events cause EXIT2 to be invoked:

- Duplicate records.
- Rerun records.

- The presence of a record type unknown to JIFSEL (a record forced at EXIT1).
- The consolidated record is to be written.

Each class of event is discussed separately. Your routine is written to accommodate each situation.

See the CA-PanAudit Plus *Installation Guide* for details on how to write your EXIT2 routine.

Read Input Exit

JIF is a two-step process:

1. Produces the consolidated file
2. CA-PanAudit Plus routines are invoked to produce the reports (see Facility Operation earlier in this chapter).

JIFRDREX is the name of the read input exit that is used to read the consolidated file. JIFRDREX reads the consolidated file, then presents a fixed-length record to CA-PanAudit Plus for reporting.

Using the Exit

JIFRDREX presents, to the routines, only the information you request (for example, job information, job and step information, job and user-appended information). Your request for information is based on parameters that are passed to JIFRDREX.

In the following table, parameters 1 through 8 indicate the information you want to process from the consolidated file.

Parameters	Options
1	Y = return job information N = job information not required
2	Y = return step information N = step information not required
3	Y = return spool information N = spool information not required
4-7	N = (N is required; these flags are reserved for future use)
8	Y = if present, return usergenerated information N = user information not required

Using the Availability Flags

JIFRDREX communicates with the routines through the first eight bytes of the record (the fields XFLAG1 through XFLAG8). These fields act as availability flags and indicate whether job, step, and spool detail are present on the record.

To indicate the level of detail, specify the appropriate parameter on the macro invocation statement of JIFREC. The numbers represent positional parameters and identify which information is requested.

JIFRDREX sets the availability flags to Y (yes) or N (no) to indicate the presence of the requested information.

XFLAG Definitions

When JIFRDREX reads records from the consolidated file, it sets the XFLAG fields (bytes 1 through 8) to indicate the record contents. The fields are defined in the following table:

For XFLAG	The Meaning/ Content Is:	JIFRDREX Exit Sets The Flags To:
1	New job information	Y = when the job information fields in a record have changed to include new information N = when the contents of the job information fields have not changed
2	New step information	Y = when step information was requested and the content of the step information fields in a record has changed N = when step information was not requested, is not present, or has been exhausted for this job
3	New spool information	Y = when spool information was requested and the contents of the spool information field in a record have changed N = when spool information was not requested, is not present, or has been exhausted for this job
4-7	Reserved for future use	N = These flags are not currently being used, and are always set to N.
8	New user	Y = when user information was requested and was present in the record N = when user information was not requested, is not present, or had been previously formatted for this job

Example

To demonstrate what is returned to the routines when processing the consolidated file, consider the following example.

Input

- JIFOPTS specifies the following parameters:

  ```
  STEPRCD=YES; SPOLRCD=YES;
  EXIT2=USEREXIT.
  ```

- USEREXIT is your routine that appends the 256-position user area to all nonorphan consolidated records.
- The consolidated job record produced from the SMF data contains information for three steps and two spooled data sets, as well as the user-added section.
- The following statement precedes the invocation of the JIF report:

  ```
  %JIFREC YYYNNNNY
  ```

- The only job record in the consolidated record file is the one described previously in item three.

Output

```
XFLAG      CONTENTS OF CONSOLIDATED RECORD FIELDS (Letters=prefixes)
FIELDS

         RECORD 1

1 = Y    CM - Accumulated job information
2 = Y    CS - Information on the first step executed in the job stream
3 = Y    CP - Information on the first spooled data set
8 = Y    User Portion - 256-byte user section

         RECORD 2

1 = N    CM - Same as in record 1
2 = Y    CS - Information on second execution in the job stream
3 = Y    CP - Information on the second spooled data set
8 = N    User Portion - Same as record 1

         RECORD 3

1 = N    CM - Same as record 1
2 = Y    CS - Information on the third step executed in the job stream
3 = N    CP - same as record 2
8 = N    User Portion - Same as record 1
```

Statistical Reporting Routines

The OS Job Information Facility (JIF) provides reporting routines and an audit report routine that produce preformatted reports from the SMF data in the consolidated file.

The routines bypass all orphan and user-modified consolidated records. (Appending a section to an otherwise unaltered consolidated record is not considered modification in this context.)

Syntax

To execute a JIF routine, you must invoke two macros. The following is the basic format:

```
              YYY    Y
%JIFREC       NNNNNNNN

%OSJIFxx      Startdate   Enddate
```

The JIFREC macro contains the file and field definitions for the contents of the consolidated file.

The Y/N (Yes/No) identifiers See the availability flags (XFLAGS) located at bytes 1 through 8 of the consolidated record. They indicate the type of information from the consolidated record that is to be made available to the JIF statistical routine.

- Y – indicates that a type of information is to be made available
- N – indicates that it is not available

See Read Input Exit earlier in this chapter.

The OSJIFxx macro identifies the reporting routines, where xx represents routines 01 through 19.

Startdate and Enddate are expressed in Julian format (YYDDD). See the selected job information reporting period for each of the routines except where noted. All parameters for all JIF reports are required.

Unless units of time are explicitly identified in a column heading of a statistical report (for example, PAGE SECONDS in report OSJIF10), all units of time shown in the reports are in minutes to four decimal places.

In columns labeled AVE ALLOCATION TIME (reports OSJIF04, OSJIF09, OSJIF11, OSJIF13), units of time are reported in seconds, to two decimal places.

Building Customized Routines

The supplied routines are general and may not always meet your needs. You can write your own routines either by modifying an existing routine or by designing new routines using CA-PanAudit Plus.

JCL Example

The following JCL shows an example which assumes that the consolidated file was created in a previous step. The DEPTTAB file is required only for the OSJIF14 routine.

```
//jobname      JOB    accounting.info
//STEPNAME     EXEC   PGM=EZTPA00
//STEPLIB      DD     ...
//SYSPRINT     DD     SYSOUT=A
//SYSUDUMP     DD     SYSOUT=A
//SYSOUT       DD     SYSOUT=A
//PANDD        DD     DSN=PAPL.macro.library,DISP=SHR
//CONSOL       DD     DSN=OSJIF.consol.idated.data,DISP=SHR
//AUDIT        DD     DSN=OSJIF.audit.data.DISP=SHR
//EZTVFM       DD     UNIT=SYSDA,SPACE=(4096,(100,100))
//SORTWK01     DD     UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK02     DD     UNIT=SYSDA,SPACE=(CYL,5)
//SORTWK03     DD     UNIT=SYSDA,SPACE=(CYL,5)
//DEPTTAB      DD     *

              DEPARTMENT TABLE

//SYSIN        DD     *
%JIFREC YNNNNNNN
%OSJIFxx yyddd yyddd
```

Statistical Routines

The following section describes available statistical routines.

Service Unit Distribution — OSJIF01

The OSJIF01 routine generates a report summarizing the service units provided in general support of all processing within the range of dates you specify. The data is sequenced by OS/390 or z/OS service unit ranges in an OS/390 or z/OS performance group.

Syntax

```
%OSJIF01  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following is an example of the OSJIF01 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF01 89354 89354
```

Output

```
                            SERVICE UNIT DISTRIBUTION (ROUTINE OSJIF01)                            PAGE      1
                          DATA SELECTED BETWEEN DATES - 89354 AND 89354

                                                          AVG      AVG      TOTAL
PER.      SERVICE        PERCENT   NUMBER     SERVICE    SRB      TCB      I/O       PAGE        SWAP
GROUP         UNITS           EXECS    OF JOBS     UNITS     TIME     TIME     COUNT    SECONDS       COUNT

0001    100000 OR GREATER    100.000       6    1,050,969    .1134    .1871              15,142          3
OPER. GROUP SUBTOTAL       100.000       6    1,050,969    .1134    .1871              15,142          3

                             100.000       6    1,050,969    .1134    .1871              15,142          3
```

Report Contents

The following describes each field name in the OSJIF01 report.

Performance Group

OS/390 or z/OS performance group.

Service Unit Types

Range of OS/390 or z/OS service units.

Percent Execs

Percentage of exec instructions performed that fall in the service unit range.

Number of Jobs

Number of jobs selected from the input file that fall in the service unit range.

Service Units

Number of service units recorded by OS/390 or z/OS for jobs that fall in the specified dates.

Average SRB Time

Average time spent under control of a Service request block (SRB) for the service unit range.

Average TCB Time

Average time spent under control of a Task control block (TCB) for the service unit range.

Total I/O Count

Total input/output requests of all jobs in the service unit range.

Page Seconds

Total number of page seconds for all jobs in the service unit range.

Swap Count

Total numbers of address space swap sequences for jobs in the service unit range.

Performance Objective Summary — OSJIF02

The OSJIF02 routine generates a summary service distribution report on SMF data recorded between the specified dates. The data is sequenced by the performance objective of your choice in a performance group.

Syntax

```
%OSJIF02  startdate  enddate  objective
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

`objective`

Any field described in the consolidated job record, can be considered as a performance objective, with the exception of the Scratch pad and working storage field areas.

Although any field can be used, most cannot produce meaningful groupings. The following is a list of suggested performance objective fields:

Field Name	Description
CMCOUNT2	Group by Julian Date (YYYDDD)
CMSYSID	Group by System ID
CMJOBNM	Group by Job Name
CMJOBPT	Group by Job Priority
CMJOBCL	Group by Job Class
CMPRGNAM	Group by Programmer Name
CMUSRID	Group by User ID

Example

The following sections show an example of the OSJIF02 report. The performance objective is Programmer Name. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF02 89354 89354 CMPRGNAM
```

Output

```
                      PERFORMANCE OBJECTIVE SUMMARY (ROUTINE OSJIF02)                   PAGE      1
                      DATA SELECTED BETWEEN DATES - 89354 AND 89354

                                   NUM. NUM.                             TOTAL                 TOTAL
PER.       PER.    PERCENT    OF      OF   SERVICE   RESIDENT     CPU     I/O      PAGE        PAGEIN
GROUP      OBJECTIVE      EXECS    JOBS    SES. UNITS      TIME      TIME    COUNT    SECONDS     PAGEOUT

0001  DATA.CENTER BKUPS  100.000       6       1,050,969   67.2843   1.8038           15,142           6
SUBTOTAL                 100.000       6       1,050,969   67.2843   1.8038           15,142           6

                         100.000       6       1,050,969   67.2843   1.8038           15,142           6
```

Report Contents

The following list describes each field name in the OSJIF02 report.

Performance Group

OS/390 or z/OS performance group.

Performance Objective

User-selected performance objective.

Percent Execs

Percentage of jobs that fall in the specified group.

Number of Jobs

Number of jobs that fall in the specified group.

Number of Sessions

TSO sessions that fall in the specified group.

Service Units

Number of service units recorded by OS/390 or z/OS for jobs that fall in the specified group.

Resident Time

Total time the transactions within the specified group remained in real storage.

CPU Time

Total combined time the transactions in the specified group spent under control of a task control block and service request block.

Total I/O Count

Total input/output requests of all jobs in the specified group.

Page Seconds

Total number of page seconds for all jobs in the specified group.

Total Page-in Page-out

Total nonswap pageins and pageouts for jobs in the specified group.

Workload Trend Analysis — OSJIF03

The OSJIF03 routine provides a workload trend analysis for a selected period of time, summarized by month.

Syntax

```
%OSJIF03  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF03 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF03 89354 89354
```

Output

```
                             WORKLOAD TREND ANALYSIS (ROUTINE OSJIF03)                         PAGE      1
                          DATA SELECTED BETWEEN DATES - 89354 AND 89354

           NUM.   NUM.                                  TOTAL
            OF      OF     ELAPSED       CPU        SERVICE       I/O         LINES     PERCENT
  MONTH    JOBS    SES.  TIME         TIME         UNITS       COUNT        PRINTED    EXECS

DECEMBER      6             71.1059     1.8038     1,050,969                              100.000
              6             71.1059     1.8038     1,050,969                              100.000
```

Report Contents

The following list describes each field name in the OSJIF03 report.

Month

The month during which the selected jobs being reported on occurred.

Number of Jobs

Number of jobs that occurred in the given month.

Number of Sessions

TSO sessions that occurred in the given month.

Elapsed Time

Total elapsed time for all jobs in the given month. This is defined as the difference between job start and end times.

CPU Time

Total combined time the transactions that occurred in the given month spent under control of a task control block and service request block.

Service Units

Number of service units recorded by OS/390 or z/OS for jobs that fall in the given month.

Total I/O Count

Total disk and tape input/output events that occurred for all jobs in the given month.

Lines Printed

Total number of lines produced by all jobs in the given month.

Percent Execs

Percentage of the total number of jobs that fall in the given month.

Performance Group Profile — OSJIF04

The OSJIF04 routine generates a summarized performance group profile report for a selected period of time and deals with average time profiles.

Syntax

```
%OSJIF04  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF04 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF04 89354 89354
```

Output

```
                              PERFORMANCE GROUP PROFILE (ROUTINE OSJIF04)                       PAGE      1
                            DATA SELECTED BETWEEN DATES - 89354 AND 89354

      NUM. NUM. AVG        AVG          AVG           AVG          AVG          AVG            AVG
PER. OF       OF     RDR     ALLOCATION    ACTIVE        RESIDENT        CPU         WRITER         ELAPSED
GROUP JOBS    SES. TIME         TIME        TIME          TIME         TIME          TIME           TIME

0001     6         500.6669     2.00         11.4660      11.2140       .3006         .0000         11.8509
         6         500.6669     2.00         11.4660      11.2140       .3006         .0000         11.8509
```

Report Contents

The following describes each field name in the OSJIF04 report.

Performance Group

OS/390 or z/OS performance group.

Number of Jobs

Number of jobs selected from the input file that fall in the service-unit range.

Number of Sessions

TSO sessions selected from the input file that fall in the performance group.

Average Reader Time

Average time each job in the performance group spent on the reader queue.

Average Allocation Time

Average allocation delay time for each job in the performance group.

Average Active Time

Average time each job in the performance group was active.

Average Resident Time

Average time each job in the performance group spent in real storage.

Average CPU Time

Average combined time each job in the selected performance group spent under control of a task control block and service request block.

Average Writer Time

Average time each job in the performance group spent in the output queue.

Average Elapsed Time

Average elapsed time for each job in the performance group.

Performance Group Summary — OSJIF05

The OSJIF05 routine provides a performance group summary report for a selected period of time and deals with resource requirements in performance groups.

Syntax

```
%OSJIF05  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF05 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF05 89354 89354
```

Output

```
                              PERFORMANCE GROUP SUMMARY (ROUTINE OSJIF05)                        PAGE      1
                             DATA SELECTED BETWEEN DATES - 89354 AND 89354

        NUM. NUM.
PER.  OF     OF    SERVICE   SWAP     TCB       SRB       CPU       ACTIVE        RESIDENT     PERCENT
GROUP   JOBS   SES. UNITS    COUNT    TIME      TIME      TIME       TIME           TIME        EXECS

0001     6          1,050,969     3     1.1231     .6807     1.8038      68.7964        67.2843   100.000
         6          1,050,969     3     1.1231     .6807     1.8038      68.7964        67.2843   100.000
```

Report Contents

The following describes each field name in the OSJIF05 report.

Performance Group

OS/390 or z/OS performance group.

Number of Jobs

Number of jobs in the performance group.

Number of Sessions

TSO sessions in the performance group.

Service Units

Number of service units recorded by OS/390 or z/OS for jobs in the selected performance group.

Swap Count

Total numbers of address space swap sequences for jobs in the performance group.

TCB Time

Total time jobs in the performance group spent under control of a task control block (TCB).

SRB Time

Total time jobs in the performance group spent under control of a service request block (SRB).

CPU Time

Total amount of CPU time used by jobs in the performance group. CPU time = TCB time + SRB time.

Active Time

Total time jobs in the performance group are active.

Resident Time

Total time jobs in the performance group spent in real storage.

Percent CPU

Percentage of total CPU time used for jobs that fall in the selected performance group.

Performance Group/Priority/Class — OSJIF06

The OSJIF06 routine provides a report that calculates timing averages for jobs categorized by job class, job priority, and performance group.

Syntax

```
%OSJIF06  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF06 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF06 89354 89354
```

Output

```
                          PERFORMANCE GROUP / PRIORITY / CLASS PROFILE (ROUTINE OSJIF06)              PAGE      1
                               DATA SELECTED BETWEEN DATES - 89354 AND 89354

       P     NUM. NUM. AVG         AVG         AVG           AVG          AVG          AVG          AVG
 P     T  C   OF     OF     RDR         WRITER      ELAPSED       TURNAROUND       SRB          TCB          CPU
 G     Y  L  JOBS   SES. TIME         TIME        TIME           TIME          TIME         TIME         TIME

0001      S   6            500.6669     .0000      11.8509         .0000        .1134        .1871        .3006
              6            500.6669     .0000      11.8509         .0000        .1134        .1871        .3006
```

Report Contents

The following describes each field name in the OSJIF06 report.

Performance Group (PG)

OS/390 or z/OS performance group.

Priority (PTY)

Priority of the job.

Class (CL)

Job class.

Number of Jobs

Number of jobs that fall in the performance group/job-class group.

Number of Sessions

TSO sessions that fall in the performance group/job-class group.

Average Reader Time

Average time each job in the performance group spent on the reader queue.

Average Writer Time

Average time each job in the performance group spent in the output queue.

Average Elapsed Time

Average elapsed time for each job in the performance group.

Average Turnaround Time

Average turnaround time for each job in the performance group.

Average SRB Time

Average time jobs in the performance group spent under control of a service request block (SRB).

Average TCB Time

Average time jobs in the performance group spent under control of a task control block (TCB).

Average CPU Time

Average combined time each job in the selected performance group spent under control of a task control block (TCB) and service request block (SRB).

Hourly Throughput Analysis — OSJIF07

The OSJIF07 routine provides an hourbyhour throughput analysis on a day-by-day basis.

Syntax

```
%OSJIF07  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF07 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF07 89354 89354
```

Output

```
                 HOURLY THROUGHPUT ANALYSIS  (ROUTINE OSJIF07)                          PAGE      1
                  DATA SELECTED BETWEEN DATES - 89354 AND 89354
                                 FOR 12/20/89

             NUM.  NUM.  AVG          AVG        AVG
  TIME        OF      OF    ACTIVE      RESIDENT      CPU       SRB          CPU        SERVICE       PAGE
 INTERVAL    JOBS     SES. TIME          TIME       TIME      TIME         TIME        UNITS      SECONDS

00 - 01 AM     6           11.4660      11.2140      .3006      .6807        1.8038    1,050,969        15,142
               6           11.4660      11.2140      .3006      .6807        1.8038    1,050,969        15,142
               6           11.4660      11.2140      .3006      .6807        1.8038    1,050,969        15,142
```

Report Contents

The following list describes each field name in the OSJIF07 report.

Time Interval

Time frame during the day in which the jobs were started.

Number of Jobs

Number of jobs that were started during the specified time interval.

Number of Sessions

TSO sessions that were started during the specified time interval.

Average Active Time

Average time each job in the time interval was active.

Average Resident Time

Average time each job in the time interval remained in real storage.

Average CPU Time

Average CPU time consumed by each job in the given time interval.

SRB Time

Total time jobs initiated in the time interval spent under control of a service request block (SRB).

CPU Time

Total CPU time consumed by each job in the given time interval.

Service Units

Number of service units recorded by OS/390 or z/OS for jobs in the selected time interval.

Page Seconds

Total number of page seconds for all jobs in the time interval.

Service Requirements — OSJIF08

The OSJIF08 routine provides a detailed service requirements report. Each job, for the selected period of time, produces a line entry on the report. Information is reported in job name sequence.

Syntax

```
%OSJIF08  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF08 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF08 89354 89354
```

Output

```
                      SERVICE REQUIREMENTS  (ROUTINE OSJIF08)                                PAGE      1
                      DATA SELECTED BETWEEN DATES - 89354 AND 89354

                    P
  JOB     PER. C   T  SERVICE      TCB       SRB     DISK I/O   TAPE I/O      PAGE      PAGEIN     PAGEOUT
  NAME    GROUP  L   Y    UNITS      TIME      TIME      COUNT      COUNT    SECONDS      COUNT       COUNT

 DBUSR053    1   S        174,180    .1783     .1078                           2,523
 DBUSR054    1   S        221,818    .2255     .1535                           3,079          5
 DBUSR055    1   S        114,786    .1441     .0738                           2,015
 DBUSR056    1   S        220,431    .2118     .1633                           2,707
 DBUSR057    1   S        108,196    .1393     .0620                           1,621
 DBUSR058    1   S        211,558    .2241     .1203                           3,197          1

                        1,050,969   1.1231     .6807                          15,142          6
```

Report Contents

The following list describes each field name in the OSJIF08 report.

Job Name

Taken from the job card.

Performance Group

OS/390 or z/OS performance group.

Class (CL)

Job class.

Priority (PTY)

Priority of the job.

Service Units

Number of service units recorded by OS/390 or z/OS for this job.

TCB Time

Time a job spent under control of a task control block (TCB).

SRB Time

Time a job spent under control of a service request block (SRB).

Disk I/O Count

Total number of disk input/output events that occurred during the execution of a job.

Tape I/O Count

Total number of tape input/output events that occurred during the execution of a job.

Page Seconds

Total number of page seconds for this job.

Page-in Count

Number of input/output page-ins recorded for this job.

Page-out Count

Number of input/output page-outs recorded for this job.

Hourly Turnaround — OSJIF09

The OSJIF09 routine generates an analysis based on the average amount of time required to process a job in a given one-hour time period. A one-page report is produced for each day that occurred in the selected time frame defined by startdate and enddate. Data is sequenced by hourly period.

Syntax

```
%OSJIF09  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF09 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF09 89354 89354
```

Output

```
                       HOURLY TURNAROUND  (ROUTINE OSJIF09)                                 PAGE      1
                  DATA SELECTED BETWEEN DATES - 89354 AND 89354
                                   FOR 12/20/89

              NUM. NUM. AVG        AVG         AVG         AVG        AVG        AVG
    TIME       OF    OF     RDR     ALLOCATION    ACTIVE     ELAPSED       CPU      WRITER      PERCENT
  INTERVAL    JOBS   SES. TIME        TIME         TIME        TIME       TIME       TIME        EXECS

 00  01 AM    6        500.6669     2.00      11.4660        11.8509    .3006       .0000    100.000
               6        500.6669     2.00       11.4660        11.8509    .3006       .0000    100.000
               6        500.6669     2.00       11.4660        11.8509    .3006       .0000    100.000
```

Report Contents

The following describes each field name in the OSJIF09 report.

Time Interval

One-hour time frame in which the job was started on the date selected.

Number of Jobs

Number of jobs started during the time interval.

Number of Sessions

TSO sessions started during the time interval.

Average Reader Time

Average time each job in the time frame spent on the reader queue.

Average Allocation Time

Average allocation delay time for each job in the time frame.

Average Active Time

Average time each job in the time frame was active.

Average Elapsed Time

Average elapsed time for each job in the time frame.

Average CPU Time

Average combined time each job in the selected time frame spent under control of a task control block (TCB) and service request block (SRB).

Average Writer Time

Average time each job in the time frame spent in the output queue.

Percent Execs

Percentage of the total number of jobs included in the report that began execution in the given time frame.

Page Peaking Periods - OSJIF10

The OSJIF10 routine produces a report in which each working day is divided into two-hour time frames, and the jobs that started during those time frames are reported on.

Syntax

```
%OSJIF10  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF10 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF10 89354 89354
```

Output

```
                         PEEK PAGING PERIODS  (ROUTINE OSJIF10)                          PAGE      1
                     DATA SELECTED BETWEEN DATES - 89354 AND 89354
                                      FOR 12/20/89

                 NUM. NUM. TOTAL        AVG        AVG
     TIME         OF     OF      I/O       ACTIVE     RESIDENT     ACTIVE    RESIDENT     PAGE         TOTAL
   INTERVAL      JOBS    SES. COUNT      TIME        TIME        TIME       TIME      SECONDS        PAGES

000:00 TO 01:59   6                        11.4660     11.2140     68.7964    67.2843    15,142             6
                  6                        11.4660     11.2140     68.7964    67.2843    15,142             6
                  6                        11.4660     11.2140     68.7964    67.2843    15,142             6
```

Report Contents

The following list describes each field name in the OSJIF10 report.

Time Interval

Two-hour time frame in which the job was started on the date selected.

Number of Jobs

Number of jobs started during the time interval.

Number of Sessions

TSO sessions started during the time interval.

Total I/O Count

Total disk and tape input/output events that occurred for all jobs that started in the given time interval.

Average Active Time

Average time each job in the time frame was active.

Average Resident Time

Average time each job in the time interval remained in real storage.

Active Time

Total time all jobs in the time frame were active.

Resident Time

Total time all jobs in the time interval remained in real storage.

Page Seconds

Total number of seconds the jobs in the time frame held a page.

Total Pages

Total number of pages used by the jobs in the time interval.

Class Structure Analysis — OSJIF11

The OSJIF11 routine produces a report that categorizes the jobs by job class. The routine assumes that the only valid classes are A through Z and 0 through 9.

Syntax

```
%OSJIF11  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

enddate

The date (YYYDDD) that the selection of job information is to end.

Example

The following sections show an example of the OSJIF11 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF11 89354 89354
```

Output

```
                    CLASS STRUCTURE ANALYSIS  (ROUTINE OSJIF11)                        PAGE      1
                  DATA SELECTED BETWEEN DATES - 89354 AND 89354

               NUMBER    AVG         AVG      AVG                AVG
      JOB        OF    ALLOCATION   ACTIVE    CPU       CPU   TURNAROUND   TAPE I/O    LINES      PERCENT
     CLASS      JOBS     TIME        TIME     TIME      TIME     TIME        COUNT     PRINTED     EXECS

CLASS S JOBS    6        2.00       11.4660   .3006    1.8038    .0000                             100.000
                6        2.00       11.4660   .3006    1.8038    .0000                             100.000
```

Report Contents

The following list describes each field name in the OSJIF11 report.

Job Class

The class in which the jobs reported on were executed.

Number of Jobs

Number of jobs selected from the input data that fall in the given job class.

Average Allocation Time

Average allocation delay time for jobs in the given job class.

Average Active Time

Average time each job in the job class was active.

Average CPU Time

Average CPU time for each job in the given class.

CPU Time

Total CPU time for all jobs in the given class.

Average Turnaround Time

Average time between the time the reader recognized the job and the time the last spooled data set (list or punch) was dispatched. If a job had no spooled output, this value will be zero in that job record.

Tape I/O Count

Total tape input/output events that occurred for all jobs in the given job class.

Lines Printed

Total number of lines produced by all jobs in the given class.

Percent Execs

Percentage of the total number of jobs that were executed in the given class.

CPU Time Distribution — OSJIF12

The OSJIF12 routine categorizes the jobs that occurred within the startdate and enddate into the following CPU time usage ranges:

```
0  -  5 seconds
5  -  30 seconds
30 -  60 seconds
1  -  2 minutes
2  -  5 minutes
5  -  10 minutes
10 -  30 minutes
30 -  60 minutes
over  1 hour
```

Syntax

```
%OSJIF12  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following is an example of the OSJIF12 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF12 89354 89354
```

Output

```
                                 CPU TIME DISTRIBUTION  (ROUTINE OSJIF12)                              PAGE      1
                              DATA SELECTED BETWEEN DATES - 89354 AND 89354

                   NUMBER              AVG     AVG     AVG                       AVG
   CPU TIME       OF JOBS      CPU     CPU     SRB     TCB      ACTIVE          ACTIVE       SERVICE     PERCENT
   INTERVAL     AND SESSIONS   TIME    TIME    TIME    TIME      TIME            TIME         UNITS        CPU

05 TO 30 SECS       6         1.8038  .3006   .1134   .1871    68.7964         11.4660    1,050,969   100.000
                    6         1.8038  .3006   .1134   .1871    68.7964         11.4660    1,050,969   100.000
```

Report Contents

The following describes each field name in the OSJIF12 report.

CPU Time Interval

CPU time usage ranges into which the jobs fell, based on CPU time.

Number of Jobs and TSO Sessions

Number of jobs and TSO sessions that fell in the given CPU time range.

CPU Time

Total CPU time for each time interval.

Average CPU Time

Average CPU time for each job in the given range.

Average SRB Time

Average time jobs in the given range spent under control of a service request block (SRB).

Average TCB Time

Average time jobs in the given range spent under control of a task control block (TCB).

Active Time

Total time all jobs in the time frame were active.

Average Active Time

Average time each job in the time range was active.

Service Units

Number of service units recorded by OS/390 or z/OS for all jobs that occurred in the given range.

Percent CPU

Percentage of all CPU time that was used by all the jobs reported on in each range.

Tape Allocation — OSJIF13

The OSJIF13 routine provides a summarized report by job name grouped by number of tapes allocated to that job.

Syntax

```
%OSJIF13  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

The following is an example of the OSJIF13 report. The report contents are described on the following page.

Input

```
%JIFREC YNNNNNNN
%OSJIF13 89354 89354
```

Output

```
                       TAPE ALLOCATION BY JOBNAME (ROUTINE OSJIF13)                              PAGE      1
                      DATA SELECTED BETWEEN DATES - 89354 AND 89354

   NUMBER              NUM.    NUM.     AVG          AVG             AVG                      DISK
    TAPES     JOB      OF        OF      ALLOCATION      ELAPSED           CPU       TAPE I/O        I/O
 ALLOCATED    NAME     JOBS       SES.     TIME          TIME           TIME       COUNT         COUNT

        1     DBUSR053    1                1.97          12.9265          .2861
        1     DBUSR054    1                2.05          13.6956          .3790
        1     DBUSR055    1                1.94           6.4291          .2179
        1     DBUSR056    1                2.04          16.8840          .3751
        1     DBUSR057    1                1.98           7.5276          .2013
        1     DBUSR058    1                2.03          13.6431          .3444
        1                 6                2.00          11.8509          .3006
                          6                2.00          11.8509          .3006
```

Report Contents

The following describes each field name in the OSJIF13 report.

Number Tapes Allocated

Number of tapes that were allocated to the job.

Job Name

Job name.

Number of Jobs

Number of times the job name being reported on occurred in the record selection dates.

Number of Sessions

Number of TSO sessions that occurred in the record selection dates.

Average Allocation Time

Average amount of allocation delay time for each execution of the job being reported on.

Average Elapsed Time

Average amount of time consumed for each execution of the job being reported, from the time it started until execution finished.

Average CPU Time

Average amount of CPU time used in each execution of the job being reported.

Tape I/O Count

Total tape input/output events that occurred during job execution.

Disk I/O Count

Total disk input/output events that occurred during job execution.

Application Trend Analysis — OSJIF14

When you invoke the OSJIF14 routine, job information is selected from the consolidated file and combined by month within a department. In the routine, the department key is the first four positions of the job name field (CMJOBNM) in the consolidated record.

This routine uses table processing to translate this key into an expanded department name.

Syntax

```
%OSJIF14  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Creating the Department Table

For OSJIF14 to translate the department keys specified in the job accounting information, a department table, called DEPTTAB, must be created as follows:

```
Positions
 1 -  4      Department key
 5 - 35      Department name
36 - 80      Unused
```

```
//DEPTTAB   DD   *

ACCOACCOUNTING
MAILMAIL ROOM
MARKMARKETING
SHIPSHIPPING & RECEIVING
                 ETC.
```

Example

The following is an example of the OSJIF14 report. The report contents are described following the output.

Input

```
%JIFREC YNNNNNNN
%OSJIF14 89353 89354
```

Output

```
                 APPLICATION TREND ANALYSIS  (ROUTINE OSJIF14)                        PAGE      1
                  DATA SELECTED BETWEEN DATES - 89353 AND 89354
                      FOR DEPARTMENT -  ACCOUNTING DEPT

          NUM. NUM.                                            TAPE
            OF    OF      ELAPSED       SRB       TCB      SERVICE       DISK         LINES    PERCENT
  MONTH   JOBS    SES.   TIME         TIME      TIME      UNITS         I/O      PRINTED      EXECS

DECEMBER            1     184.0383      .0095     .0515        27,005                                .208
                    1     184.0383      .0095     .0515        27,005                                .208

             APPLICATION TREND ANALYSIS  (ROUTINE OSJIF14)                            PAGE      2
             DATA SELECTED BETWEEN DATES - 89353 AND 89354
                  FOR DEPARTMENT -  ADMINISTRATION

          NUM. NUM.                                            TAPE
            OF    OF      ELAPSED       SRB       TCB      SERVICE       DISK         LINES    PERCENT
  MONTH   JOBS    SES.   TIME         TIME      TIME      UNITS         I/O      PRINTED      EXECS

DECEMBER          15      16.5209      .3350     .9178       590,262                      132      3.125
                  15      16.5209      .3350     .9178       590,262                      132      3.125
```

```
                        APPLICATION TREND ANALYSIS  (ROUTINE OSJIF14)                       PAGE      3
                         DATA SELECTED BETWEEN DATES - 89353 AND 89354
                             FOR DEPARTMENT -  ACQUISITION DEPT

           NUM. NUM.                                               TAPE
             OF     OF      ELAPSED      SRB      TCB      SERVICE      DISK        LINES    PERCENT
  MONTH    JOBS     SES.   TIME        TIME     TIME      UNITS        I/O      PRINTED     EXECS

DECEMBER       1             5.9751      .1348     .2056      142,952       147        9,122       .208
               1             5.9751      .1348     .2056      142,952       147        9,122       .208

   .       .   .         .        .      .        .        .          .       .
   .       .   .         .        .      .        .        .          .       .
   .       .   .         .        .      .        .        .          .       .

                        APPLICATION TREND ANALYSIS  (ROUTINE OSJIF14)                       PAGE     49
                         DATA SELECTED BETWEEN DATES - 89353 AND 89354
                             FOR DEPARTMENT - UNKNOWN DEPT. = YODE

           NUM. NUM.                                               TAPE
             OF     OF      ELAPSED      SRB      TCB      SERVICE      DISK        LINES    PERCENT
  MONTH    JOBS     SES.   TIME        TIME     TIME      UNITS        I/O      PRINTED     EXECS

DECEMBER        2             6.2008      .0401      .8046     462,897                                .416
                2             6.2008      .0401      .8046     462,897                                .416

              443     37   7,658.4790    13.6387    50.2666   33,659,840     6,478       31,811    100.000
```

Report Contents

The following describes each field name in the OSJIF14 report.

Department

In-house department title, taken from DEPTTAB.

Month

Period being reported on, taken from MONTTAB.

Number of Jobs

Number of jobs that fell in the department and date being reported on.

Number of Sessions

Number of TSO sessions that fell in the department and date being reported on.

Elapsed Time

Total elapsed time for all jobs in the given period. This is defined as the difference between job start and end times.

SRB Time

Time the jobs spent under control of a service request block (SRB).

TCB Time

Time the jobs spent under control of a task control block (TCB).

Service Units

Number of service units recorded by OS/390 or z/OS for all jobs in the group.

Tape/Disk I/O Count

Total disk and tape input/output events that occurred for all jobs in the given department.

Number of Lines

Total number of lines produced by all jobs in the given department.

Percent Execs

Percentage of the total number of jobs in the given department that are represented by the selected data.

Abends by Abend Code — OSJIF15

The OSJIF15 routine produces a report grouped by abend code that lists the job and step names which abnormally terminated during the period being reported on.

Syntax

```
%OSJIF15  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

On the following pages is an example of the OSJIF15 report. The report contents are described following the output.

Input

```
%JIFREC YYNNNNNN
%OSJIF15 89353 89354
```

Output

```
                     ABENDS BY ABEND CODE   (ROUTINE OSJIF15)                         PAGE      1
                 DATA SELECTED BETWEEN DATES - 89353 AND 89354

                                               STEP       STEP                      STEP          STEP
            ABEND  PROGRAM     JOB      STEP     RUN      START       USER        ELAPSED        CPU
             CODE    NAME      NAME     NAME     DATE      TIME    INFORMATION     TIME         TIME

            S0C4   TESTPROG  SCHMIDJ  TESTPROG  89/17/33  36:17:33  -               340.0000    34.0000
                   TESTPROG  SCHMIDJ  TESTPROG  89/17/35  53:17:35  -               231.0000    33.0000
ABEND CODE                                                                          571.0000    67.0000
            S222   UTL120    JAMESM   LOG       89/17/38  35:17:38  -               230.0000    10.0000
                   IEBGENER  JAMESM   GENER1    89/17/38  37:17:38  -               493.0000    38.0000
                   IEBGENER  JAMESM   GENER2    89/17/38  40:17:38  -               713.0000    32.0000
                   IFOX00    JAMESM   ASM       89/17/38  44:17:39  -             2,603.0003    10.0002
                   IEBUPDTE  JAMESM   UPDATE    89/17/39  00:17:39  -               601.0000    32.0000
                   UTL120    JAMESM   LOG       89/17/39  04:17:39  -               261.0000    10.0000
                   IEBGENER  JAMESM   GENER1    89/17/39  06:17:39  -               368.0000    30.0000
                   IEBGENER  JAMESM   GENER2    89/17/39  08:17:39  -               633.0000    30.0000
                   IFOX00    JAMESM   ASM       89/17/39  12:17:39  -             2,785.0003     8.0002
                   IEBUPDTE  JAMESM   UPDATE    89/17/39  29:17:39  -               548.0000    28.0000
                   UTL120    JAMESM   LOG       89/17/51  19:17:51  -               246.0000    10.0000
                   IEBGENER  JAMESM   GENER1    89/17/51  21:17:51  -               233.0000    26.0000
                   IEBGENER  JAMESM   GENER2    89/17/51  22:17:51  -               511.0000    28.0000
                   IFOX00    JAMESM   ASM       89/17/51  25:17:51  -             5,013.0004    26.0003
                   IEBUPDTE  JAMESM   UPDATE    89/17/51  56:17:51  -               485.0000    26.0000
                   IKJEFT01  SCHEDUL  PANSOPHI  89/22/35  13:22:38  -             9,266.0006    16.0005
                   IEV90     SCHMIDJ  ASM       89/17/35  45:17:35  -               890.0000    39.0000
                   IEWL      SCHMIDJ  LKED      89/17/35  50:17:35  -               425.0000    24.0000
                   TMSAUDIT  DAYUCC1A STEPA     89/07/49  51:07:50  -             9,600.0001     4.0000
                   TMSCOPY   DAYUCC1A STEP1     89/07/50  48:07:52  -             2,685.0006    91.0004
                   TMSEXPDT  DAYUCC1A STEP1A    89/07/52  05:07:52  -             3,760.0002    59.0001
                   TMSCYCLE  DAYUCC1A STEP2     89/07/52  27:07:52  -             3,040.0001    86.0001
                   TMSCTLG   DAYUCC1A STEP3     89/07/52  46:07:53  -             6,863.0004    62.0003
                   TMSCLEAN  DAYUCC1A STEP4     89/07/53  27:07:54  -             5,650.0004     3.0002
                   TMSRPT2   DAYUCC1A UCC1TMS   89/07/54  01:07:54  -             4,993.0004    96.0001
                   TMSRPT3   DAYUCC1A UCC1TMS   89/07/54  31:07:54  -             3,790.0002    23.0001
                   TMSRPT4   DAYUCC1A UCC1TMS   89/07/54  54:07:55  -             3,140.0001    88.0000
                   TMSRPT6   DAYUCC1A UCC1TMS   89/07/55  13:07:55  -             3,378.0002     9.0001
                   TMSCLEAN  DAYUCC1A STEP11    89/07/55  33:07:55  -             2,696.0001    76.0000
                   PAN#2     FLAHERTV BACKUP    89/08/18  47:08:30  -             6,501.0014    92.0009
                   PAN#2     FLAHERTV BACKUP    89/08/30  27:08:34  -             6,550.0008    78.0006
                   FDRDSF    DBUSR054 DBFDR     89/00/01  27:00:13  -             5,033.0040     1.0023
                   IKJEFT01  LUTE     PRODLMP   89/07/22  35:15:30  -             2,911.0032    89.0028
                   IKJEFT01  LUTE2    PRODLMP   89/07/22  35:13:27  -             7,798.0138      .0117
                   IKJEFT01  PEARSON  PEARSON1  89/07/31  08:16:33  -             6,750.0253    78.0226
                   IKJEFT01  COVAS    COVAS     89/07/40  03:08:00  -             7,013.0061    99.0051
                   JTVMMESG  PEARSONC INITMSG   89/07/40  26:07:40  -                61.0000     6.0000
                   PANEXEC   PEARSONC LCSCHKO   89/07/40  27:07:41  -             7,443.0006    88.0006
                   JTVMMESG  PEARSONC SCCSMSG   89/07/41  11:07:41  -                 23.0000     5.0000
                   JTVMMESG  PEARSONC FAILMSG   89/07/41  12:07:41  -                5.0000      .0000
                   JTVMMESG  PEARSONC ABNDMSG   89/07/41  12:07:41  -                 5.0000      .0000
                   JTVMMESG  PEARSONC INITMSG   89/07/41  12:07:41  -                30.0000     6.0000
```

```
PANEXEC   PEARSONC  LCSCHKO   89/07/41  13:07:41  -        7,693.0006   91.0006
JTVMMESG  PEARSONC  SCCSMSG   89/07/41  59:07:41  -           46.0000    5.0000
JTVMMESG  PEARSONC  FAILMSG   89/07/41  59:07:41  -            5.0000     .0000
JTVMMESG  PEARSONC  ABNDMSG   89/07/41  59:07:41  -            8.0000     .0000

  .     .     .     .     .     .     .                  .       .
  .     .     .     .     .     .     .                  .       .
  .     .     .     .     .     .     .                  .       .

CMFBATCH  PEARSON   PVDEL1    89/09/08  23:09:08  -        1,730.0001    3.0000
CMFBATCH  PEARSON   PVADD2    89/09/08  34:09:08  -        1,566.0000   96.0000
CMFBATCH  PEARSON   UPDCF3    89/09/08  43:09:08  -        2,396.0002   28.0002
IKJEFT01  PEARSON   ENDMSG    89/09/08  58:09:08  -            6.0000     .0000
IKJEFT01  PEARSON   FAILMSG   89/09/08  58:09:09  -        2,383.0001   39.0001
                                                         664,287.5991  675.4644
```

Report Contents

The following describes each field name in the OSJIF15 report.

Abend Code

System or user-issued abend code.

Program Name

Name of the program that ended abnormally.

Job Name

Job name.

Step Name

Step name.

Step Run Date

Date on which the abended step began execution.

Step Start Time

Time of day that the abended step began execution.

User Information

Blank if no user information present.

Step Elapsed Time

Length of time between the time the step began execution and the time it abended.

Step CPU Time

Amount of CPU time consumed by the step execution prior to abending.

Abend by Program — OSJIF16

The OSJIF16 routine provides the same information as OSJIF15, with the exception that in OSJIF16 the information is presented by program name.

The report content explanations for this routine are identical to those listed in the OSJIF15 routine.

Syntax

```
%OSJIF16  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

On the following pages is an example of the OSJIF16 report. The report contents are described in the OSJIF15 output.

Input

```
%JIFREC YYNNNNNN
%OSJIF16 89353 89354
```

Output

```
                    ABNORMAL TERMINATIONS BY PROGRAM  (ROUTINE OSJIF16)                            PAGE      1
                     DATA SELECTED BETWEEN DATES - 89353 AND 89354

                                                       STEP      STEP                    STEP           STEP
          PROGRAM      JOB     ABEND     STEP          RUN       START        USER       ELAPSED         CPU
            NAME       NAME     CODE      NAME         DATE       TIME     INFORMATION   TIME            TIME

          AMASPZAP   SCHMIDJZ   SA03     STEP1        89/13/32   46:13:32  -                775.0000      34.0000
PROGRAM                                                                                     775.0000      34.0000

          AMDPRDMP   MCREYNO1   SA03     PRDUMP2      89/11/51   09:11:52  -              7,700.0016      91.0013
                     MCREYNO1   SA03     PRDUMP2      89/15/10   09:15:11  -              2,875.0019      29.0014
PROGRAM                                                                                  10,575.0035     120.0027

          APCS5102   COVAS2     SA03     APCS5102     89/08/25   42:08:26  -              3,238.0002      37.0002
                     COVAS2     SA03     APCS5102     89/09/11   39:09:12  -              3,941.0002      42.0002
PROGRAM                                                                                   7,179.0004      79.0004
```

```
          APCS5320  COVAS1    SA03    APCS5320  89/14/26  04:14:26  -         8,606.0004      71.0004
                    COVAS1    SA03    APCS5320  89/14/32  40:14:33  -         8,570.0004      55.0004
                    COVAS1    S222    APCS5320  89/14/36  06:14:39  -         4,201.0046      24.0037
                    COVAS1    S222    APCS5320  89/14/45  01:14:49  -         5,176.0036      46.0030
                    COVAS1    SA03    APCS5320  89/14/50  37:14:51  -         9,605.0005       8.0004
                    COVAS1    S013    APCS5320  89/15/37  51:15:38  -         6,741.0004      41.0004
                    COVAS1    SA03    APCS5320  89/15/40  32:15:41  -         4,961.0004      13.0003
                    COVAS1    S013    APCS5320  89/15/42  35:15:43  -         2,093.0004      24.0003
                    COVAS1    S222    APCS5320  89/15/47  12:15:50  -         3,060.0054      88.0043
                    COVAS1    S013    APCS5320  89/15/53  04:15:53  -         5,773.0004      26.0003
                    COVAS1    S013    APCS5320  89/15/54  26:15:55  -         7,630.0004      70.0004
PROGRAM                                                                      66,416.0169     466.0139

          APCS5391  COVAS1    SA03    APCS5391  89/15/55  12:15:55  -             6.0000        .0000
                    COVAS1    SA03    APCS5391  89/15/53  39:15:53  -             8.0000        .0000
                    COVAS1    SA03    APCS5391  89/15/43  48:15:43  -             5.0000        .0000
                    COVAS1    SA03    APCS5391  89/15/41  02:15:41  -           170.0000        .0000
                    COVAS1    SA03    APCS5391  89/15/38  32:15:38  -             6.0000        .0000
                    COVAS1    SA03    APCS5391  89/14/51  35:14:51  -           223.0000        .0000
                    COVAS1    SA03    APCS5391  89/14/33  32:14:33  -           285.0000        .0000
                    COVAS1    SA03    APCS5391  89/14/26  56:14:26  -           171.0000        .0000
                                                                            874.0000        .0000
  .      .    .     .       .      .      .  .         .       .
  .      .    .     .       .      .      .  .         .       .
  .      .    .     .       .      .      .  .         .       .

PROGRAM

          IFOX00    LUTE1     SA03    ASM       89/10/59  51:11:00  -         2,451.0001      44.0001
                    JAMESM    SA03    ASM       89/16/29  03:16:29  -         4,530.0001      67.0001
                    JAMESM    SA03    ASM       89/16/32  20:16:32  -         3,541.0003      21.0002
                    JAMESM    SA03    ASM       89/16/34  14:16:34  -         6,526.0003      58.0003
                    JAMESM    SA03    ASM       89/16/22  14:16:22  -         1,966.0000      61.0000
                    JAMESM    SA03    ASM       89/16/23  36:16:23  -         3,841.0002      82.0002
                    JAMESM    SA03    ASM       89/16/25  39:16:25  -         1,253.0000      62.0000
                    JAMESM    SA03    ASM       89/16/26  09:16:26  -         2,261.0001      19.0001
                    JAMESM    SA03    ASM       89/16/27  30:16:27  -         2,726.0002      76.0002
                    JAMESM    SA03    ASM       89/16/28  18:16:28  -         4,526.0003      11.0002
                    DIRADSNT  SA03    ASM       89/17/07  05:17:10  -         7,830.0061      21.0056
                    JAMESM    SA03    ASM       89/17/00  11:17:00  -         2,735.0003      13.0002

                                                                        664,287.5991*   9675.4644*
```

Abend by Programmer — OSJIF17

The OSJIF17 routine provides the same information as OSJIF15 and OSJIF16 with the exception that in OSJIF17 the information is presented in order by programmer name.

The report content explanations for this routine are identical to those listed in the OSJIF15 routine.

Syntax

```
%OSJIF17  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

On the following pages is an example of the OSJIF17 report. The report contents are described in the OSJIF15 output.

Input

```
%JIFREC YNNNNNNN
%OSJIF17 89353 89353
```

Output

```
                        ABEND BY PROGRAMMER  (ROUTINE OSJIF17)                          PAGE     1
                      DATA SELECTED BETWEEN DATES - 89353 AND 89353

                                                    STEP     STEP        STEP          STEP
             PROGRAMMER   JOB      PROGRAM   ABEND  STEP    RUN      START       ELAPSED       CPU
               NAME       NAME       NAME    CODE   NAME    DATE     TIME        TIME          TIME

        ADMIN            PRJPANBK   PAN#2     SA03   STEP1    89/17/59   04:18:00    8,588.0008     84.0006
                         PRJPANBK   PAN#2     SA03   STEP2    89/18/00   55:18:01    8,853.0003     19.0002
                         PRJPANBK   PAN#2     SA03   STEP3    89/18/01   48:18:01       93.0000       .0000
PROGRAMMER                                                                              17,534.0011    103.0008

        AL TREVINO       TREVINO1   JTVMMESG  SA03   INITMSG  89/09/24   35:09:24      255.0000      6.0000
                         TREVINO1   PANEXEC   SA03   LCSASMA  89/09/24   37:09:28    9,730.0023     58.0021
                         TREVINO1   JTVMMESG  SA03   SCCSMSG  89/09/28   35:09:28       31.0000      5.0000
                         TREVINO1   JTVMMESG  SA03   FAILMSG  89/09/28   35:09:28        5.0000       .0000
                         TREVINO1   JTVMMESG  SA03   ABNDMSG  89/09/28   36:09:28        3.0000       .0000
PROGRAMMER                                                                            10,024.0023     69.0021

        ARRAYCOMP        KADLCOMP   PAN#1     SA03   PANSTEP  89/10/26   41:10:26      636.0000     40.0000
PROGRAMMER                                                                               636.0000     40.0000
```

```
         ARR007        KINSR007   PAN#1      SA03    PANSTEP  89/11/22   22:11:22        598.0000       31.0000
PROGRAMMER                                                                               598.0000       31.0000

         ARR008        KINSR008   PAN#1      SA03    PANSTEP  89/13/52   20:13:52      2,170.0000       39.0000
PROGRAMMER                                                                             2,170.0000       39.0000

         A2CICS17      DIRADSNT   PAN#1      SA03    PAN      89/17/06   31:17:07      5,553.0001       52.0001
                       DIRADSNT   IFOX00     SA03    ASM      89/17/07   05:17:10      7,830.0061       21.0056
                       DIRADSNT   IEBUPDTE   SA03    BLDMBR   89/17/10   52:17:10      1,166.0000       61.0000
                       DIRADSNT   IEWL       SA03    LNKEDT   89/17/10   59:17:11      1,796.0000       50.0000
PROGRAMMER                                                                            16,345.0062      184.0057

         CHILDS        CHILDSMS   PAN#8      SA03    STEP1    89/11/58   12:11:59      8,181.0010       83.0009
PROGRAMMER                                                                             8,181.0010       83.0009

         CLEANUP       GRYZIK     PAN#2      SA03    STEP1    89/10/22   31:10:22      2,876.0001       44.0001
PROGRAMMER                                                                             2,876.0001       44.0001

         COPY EXCHANGE SYSEXCH    IEBGENER   SA03    STEP1    89/10/17   22:10:20      7,075.0000       29.0000
                       SYSEXCH    IEBGENER   SA03    STEP2    89/10/20   04:10:20      5,078.0000       26.0000
                                                                                      12,153.0000       55.0000

   .       .          .       .     .     .       .      .          .         .
   .       .          .       .     .     .       .      .          .         .
   .       .          .       .     .     .       .      .          .         .

         012AD2TRAN    KINSTRAN   PAN#1      SA03    PANSTEP  89/10/15   41:10:15        828.0000       47.0000
PROGRAMMER                                                                               828.0000       47.0000

         021AXXXX      HARTXXXX   PAN#1      SA03    PANSTEP  89/10/35   01:10:35      1,071.0000       99.0000
PROGRAMMER                                                                             1,071.0000       99.0000

         065DDOSA      IWANDOSA   PAN#1      SA03    PANSTEP  89/11/06   29:11:06      1,101.0001       27.0001
                       IWANDOSA   PAN#1      SA03    PANSTEP  89/11/02   27:11:02        820.0000       93.0000
PROGRAMMER                                                                             1,921.0001      120.0001

         065DVICKLS    IWANCKLS   PAN#1      SA03    PANSTEP  89/15/39   09:15:39        845.0000       48.0000
PROGRAMMER                                                                               845.0000       48.0000
                                                                                     150,163.3958*   2369.2953*
```

Audit File Statistical Report — OSJIF18

The OSJIF18 routine provides the same accounting report as JIFSEL does, with the exception that OSJIF18 reports on all records in the audit file.

Syntax

```
%OSJIF18
```

Example

The following is an example of the OSJIF18 report. For an explanation of the report contents, see JIFSEL Data Sets and Audit Report earlier in this chapter.

Input

```
%OSJIF18
```

Output

```
                             COMPUTER ASSOCIATES - OS JOB INFORMATION FACILITY         8.58.03
    PART ONE ** INPUT **

     -A-    DATASET PROCESSING       DATASET NAME      SUGEL.TEST.SMFDATA
                                     VOLUME SERIAL     USR011

     -B-     DATE/TIME               FIRST SMF RECORD       12/20/89  00:51:04
                                     LAST  SMF RECORD       12/20/89  00:55:03

     -C-     RECORD TOTALS           SMF RECORDS READ           22762
                                     SMF RECORDS REJECTED       19365
                                     SMF RECORDS FORCED             0
                                     SMF RECORDS DUPLICATE          0

     -D-     # I.P.L. RECORDS               0

     -E-    # DATA LOST RECORDS             0

        PART TWO ** OUTPUT **

     -A-    RECORD  TOTALS       CONSOLIDATED RECORDS  CREATED       854
                                                       DELETED         0
                                                       ORPHAN        373
                                                       RERUN           1
                                                       MODIFIED        0

     -B-    DATE/TIME            FIRST CONSOLIDATED RECORD     12/18/89  23:55:50
                                 LAST CONSOLIDATED RECORD      12/19/89  22:35:13
```

TSO Session Analysis — OSJIF19

The OSJIF19 routine does an analysis of TSO sessions by date in the range selected. This routine reports on TSO sessions only.

Syntax

```
%OSJIF19  startdate  enddate
```

`startdate`

The date (YYYDDD) that the selection of job information is to begin.

`enddate`

The date (YYYDDD) that the selection of job information is to end.

Example

On the following pages is an example of the OSJIF19 report. The report contents are described following the output.

Input

```
%JIFREC YNNNNNNN
%OSJIF19 89353 89354
```

Output

```
                       TSO SESSION ANALYSIS  (ROUTINE OSJIF19)                          PAGE     1
                    DATA SELECTED BETWEEN DATES - 89353 AND 89354
                                    LOGON DATE: 12/19/89

                                                AVG                   PCT     AVG                  AVG
 JOB     START     STOP   TSU     CONNECT     CONNECT     ACTIVE       OF     ACTIVE      CPU      CPU       TSO       TSO
 NAME     TIME      TIME   NO.   TIME         TIME        TIME      TOTAL    TIME       TIME      TIME      PUTS      GETS

ADAMS    10:34:46 12:59:30  316     144.7213    144.7213       .9697   .411       .9697     .0341    .0341    160,000
        16:34:55 17:05:291,017      30.5555     30.5555      .5652   .240       .5652     .0288    .0288    120,000
JOBNAME TOTALS              2     175.2768     87.6384      1.5349   .651       .7674     .0629    .0314    280,000

BANYARD   8:52:05 11:56:08   95     184.0383    184.0383      1.3000   .552      1.3000     .0610    .0610    340,000
JOBNAME TOTALS              1     184.0383    184.0383      1.3000   .552      1.3000     .0610    .0610    340,000

BAKER    10:10:34 10:15:59  232       5.4281      5.4281       .0963   .040       .0963     .0074    .0074     10,000
JOBNAME TOTALS              1       5.4281      5.4281       .0963   .040       .0963     .0074    .0074     10,000

CHATMAN  11:02:51 13:28:40  379     145.8135    145.8135      4.6227  1.963      4.6227     .4094    .4094    790,000
        13:28:54 16:43:58  558     195.0801    195.0801     10.6600  4.527     10.6600    1.2063   1.2063    840,000
JOBNAME TOTALS              2     340.8936    170.4468     15.2827  6.491      7.6413    1.6157    .8078  1,630,000

DRAKE     7:40:03  8:00:45   19      20.7028     20.7028      7.8050  3.315      7.8050     .6199    .6199    770,000
         8:00:54 12:16:46   34     255.8643    255.8643     20.6529  8.772     20.6529    1.9987   1.9987  2,520,000
        13:28:43 16:03:28  557     154.7503    154.7503     24.4115 10.369     24.4115    1.4736   1.4736  3,690,000
JOBNAME TOTALS              3     431.3174    143.7724     52.8694 22.456     17.6231    4.0922   1.3640  6,980,000

EDISON   13:50:09 13:52:44  582       2.5681      2.5681      1.5468   .657      1.5468     .0376    .0376     80,000
JOBNAME TOTALS              1       2.5681      2.5681      1.5468   .657      1.5468     .0376    .0376     80,000

FISH      9:20:55  9:59:13  135      38.3033     38.3033      1.0702   .454      1.0702     .0456    .0456     30,000
JOBNAME TOTALS              1      38.3033     38.3033      1.0702   .454      1.0702     .0456    .0456     30,000

GRAVES   10:00:35 12:27:38  207     147.0578    147.0578      2.1470   .911      2.1470     .1613    .1613    300,000
        12:58:09 16:33:24  519     215.2548    215.2548      7.3301  3.113      7.3301     .4261    .4261  1,170,000
JOBNAME TOTALS              2     362.3126    181.1563      9.4771  4.025      4.7385     .5874    .2937  1,470,000

JONES     9:37:14 17:20:49  163     463.5750    463.5750     19.9872  8.489     19.9872    2.0402   2.0402 31,220,000
JOBNAME TOTALS              1     463.5750    463.5750     19.9872  8.489     19.9872    2.0402   2.0402 31,220,000

LARKIN   15:08:13 17:34:18  698     146.0848    146.0848      1.5589   .662      1.5589     .0911    .0911    310,000
JOBNAME TOTALS              1     146.0848    146.0848      1.5589   .662      1.5589     .0911    .0911    310,000

LARKIN1  12:16:07 15:01:44  463     165.6090    165.6090      1.4115   .599      1.4115     .1089    .1089    110,000
JOBNAME TOTALS              1     165.6090    165.6090      1.4115   .599      1.4115     .1089    .1089    110,000

MADISON  13:53:01 14:22:36  586      29.5795     29.5795       .8550   .363       .8550     .0586    .0586    280,000
JOBNAME TOTALS              1      29.5795     29.5795       .8550   .363       .8550     .0586    .0586    280,000

MERRILL   7:22:35 15:30:52    6     488.2928    488.2928      6.9268  2.942      6.9268     .3289    .3289  1,750,000
JOBNAME TOTALS              1     488.2928    488.2928      6.9268  2.942      6.9268     .3289    .3289  1,750,000
```

```
NORTON    7:22:35 13:27:22   12     364.7820    364.7820      17.4158  7.397     17.4158     1.3800   1.3800  7,280,000
         13:27:36 15:23:41  551     116.0895    116.0895      8.9512  3.802      8.9512      .4555    .4555  1,410,000
JOBNAME TOTALS                2     480.8715    240.4357      26.3670 11.199     13.1835     1.8355    .9177  8,690,000

 .      .     .     .     .       .          .  .       .      .     .     .
 .      .     .     .     .       .          .  .       .      .     .     .
 .      .     .     .     .       .          .  .       .      .     .     .

WALACE    8:05:47 10:43:46   44     157.9838    157.9838        .7664   .325       .7664      .0539    .0539     50,000
JOBNAME TOTALS                1     157.9838    157.9838        .7664   .325       .7664      .0539    .0539     50,000

ZEPHR     8:21:02 16:41:23   57     500.3430    500.3430      12.5884  5.347     12.5884      .6356    .6356  2,540,000
JOBNAME TOTALS                1     500.3430    500.3430      12.5884  5.347     12.5884      .6356    .6356  2,540,000

DATE TOTALS                  37   6,260.5120    169.2030     235.4270   .000*     6.3628    17.1476    .4634 67,580,000
                            37   6,260.5120    169.2030     235.4270   .000*     6.3628    17.1476    .4634 67,580,000
```

Report Contents

The following describes each field name in the OSJIF19 report.

Job Name

The TSO session name.

Start Time

Time the session started (on the 24-hour clock).

Stop Time

Time the session ended (on the 24-hour clock).

TSO No.

The TSO session number assigned at logon time.

Connect Time

Length of time the user was connected.

Average Connect Time

Average length of time the user was connected.

Active Time

Total length of time the TSO user was active.

PCT. of Total

Percentage of total time this TSO session was active.

Average Active Time

Average length of time the TSO user was active.

CPU Time

Number of CPU minutes used during the TSO session.

Average CPU Time

Average number of CPU minutes used during the TSO session.

TSO TPUTS

Number of TPUTS completed during the TSO session.

TSO TGETS

Number of TGETS completed during the TSO session.

Weighting and Costing Examples

Weighting and costing examples contain sample CA-PanAudit Plus routines that demonstrate the use of the CA-Easytrieve Plus macro facility to generate weighting, costing, and billing information from the SMF data in the consolidated file.

Seven job accounting examples are shown on the following pages:

- Two for weighting:
  - Job Priority Weighting – EXWGTPTY
  - CPU Weighting – EXWGTCPU
- Five for costing:
  - CPU Charging – EXCHGCPU
  - I/O Charging – EXCHGIO
  - TPUT and TGET Charging – EXCHGTPG
  - Unit Record Device Charging – EXCHGUR
  - Combined Costs – EXCOSTS

These routines are supplied with the JIF and stored in your CA-PanAudit Plus macro library. They may be used without change or used as models for creating your own algorithms and routines.

Seven example CA-PanAudit Plus routines are shown later in this chapter to give you a feel for how billing routines can be created from the SMF data. The routines are:

- Invoice Ledger–JIFBEX01
- Detail Charge Audit–JIFBEX02
- Job Charge Summary–JIFBEX03
- Data Center Cost Recovery–JIFBEX04
- Monthly Utilization by Cost Center–JIFBEX05
- Data Processing Invoice–JIFBEX06
- TSO User Charging Report–JIFBEX07

Note: These examples are for demonstration only.

Building Customized Routines

Any billing algorithm must be unique to the installation and company. This allows the user to alter the parameters of costs to suit changing overheads, hardware configurations, and other factors. For example, the actual charges vary, including the costs of tape versus disk I/O, core costs, unit amounts, CPU utilization, or execution class.

In addition, an installation may run a priority scheme. For example, a priority seven or higher job costs twice the basic amount, and priority nine and above cost three times the basic amount.

Also, in a multi-CPU environment, weighted charges may be required to control the cost of one second of CPU time on machines with differing speeds.

Considering these factors, there can be no universal definitive costing algorithm.

Job Priority Weighting — EXWGTPTY

The EXWGTPTY routine compares the execution priority field from the job portion of the consolidated record (CMJOBPT) against a priority, then fills a work field (CXWGTPTY) with the weighting factor. CXWGTPTY is used in a later calculation to develop a cost.

Syntax

```
%EXWGTPTY     comparison   priority   weight
```

`comparison`

Indicates any valid CA-PanAudit Plus relational operator:

```
EQ  =             EQUAL TO
NE  ¬=   NQ       NOT EQUAL TO
LT  <    LS       LESS THAN
LE  <=   LQ  ¬>   LESS THAN OR EQUAL TO
GT  >    GR       GREATER THAN
GE  >=   GQ  ¬<   GREATER THAN OR EQUAL TO
```

`priority`

Indicates the priority that is to be tested.

`weight`

Indicates the weighting factor to be used if the condition tested for is true.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMJOBPT &OPERATOR &PRIORITY
   CXWGTPTY = &WEIGHT
END-IF
```

This routine must follow record qualification logic and precede cost calculation.

Example

```
%EXWGTPTY LE  7 1
%EXWGTPTY GQ  8 2
%EXWGTPTY GQ 10 2.5
```

In this example, the contents of CXWFGTPTY are 1.00 for jobs with a priority of 7 or less, 2.00 for jobs with a priority of 8 or 9, and 2.50 for jobs with a priority of 10 or higher.

CPU Weighting — EXWGTCPU

The EXWGTCPU routine compares the system ID field (CMSYSID) from the consolidated record with the user-supplied SYSID parameter for an equal condition. If the comparison is true, then the weight parameter is moved to a work field (CXWGTCPU). CXWGTCPU is used in a later calculation to develop cost.

Syntax

```
%EXWGTCPU     sysid   weight
```

`sysid`

A fourposition alphanumeric constant that is to be compared to the system ID field (CMSYSID) in the consolidated record.

`weight`

The weighting to be used if the condition tested for is true.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMSYSID = '&SYSID'
   CXWGTCPU = &WEIGHT
END-IF
```

This routine must follow record qualification logic and precede cost calculation.

Example

```
%EXWGTCPU H158 1.0
%EXWGTCPU 4341 1.5
```

In this example, jobs that executed on the 370/158 are weighted with a factor of 1.0 which represents basic cost. Jobs executing on the 4341 are weighted with a factor of 1.5 or one and one-half times the basic cost. This weighting factor is stored in CXWGTCPU.

CPU Charging — EXCHGCPU

The EXCHGCPU routine performs two types of CPU utilization calculations:

- A calculation to generate a cost for each minute of service request block (SRB) CPU utilization
- A calculation to generate a cost for each minute of task control block (TCB) CPU utilization.

Both these charges are weighted by the CPU and priority weight factor fields. This routine yields three cost fields:

```
CXCPUSRB = SRB CPU Utilization
CXCPUTCB = TCB CPU Utilization
CXCPUCST = CXCPUSRB + CXCPUTCB
```

Syntax

```
%EXCHGCPU     srbcost   tcbcost
```

`srbcost`

The charge unit for each minute of CPU utilization under a service request block (SRB).

`tcbcost`

The charge unit for each minute of CPU utilization under a task control block (TCB).

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
* CPU CHARGING ALGORITHM
*  ARE WE PROCESSING STEPS
IF XFLAG2 = 'X'
   CXCPUSRB = CSSRB * &SRBCOST * CXWGTCPU * CXWGTPTY
   CXCPUTCB = CSTCB * &TCBCOST * CXWGTCPU * CXWGTPTY
   CXCPUCST = CXCPUSRB + CXCPUTCB
ELSE
   CXCPUSRB = CMTMSRB * &SRBCOST * CXWGTCPU * CXWGTPTY
   CXCPUTCB = CMTMTCB * &TCBCOST * CXWGTCPU * CXWGTPTY
   CXCPUCST = CXCPUSRB + CXCPUTCB
END-IF
```

Note: It is your responsibility to move an X to the field XFLAG2 before including the example CPU charging routine, if the CPU costs are being developed from steps.

Example

```
%EXCHGCPU 7.45 9.9
```

In this example, SRB costs are developed at a rate of 7.45 charge units per minute of SRB time, and TCB costs are developed at a rate of 9.90 charge units per minute of TCB time.

I/O Charging — EXCHGIO

The EXCHGIO routine develops I/O usage charges. This routine yields seven fields:

- CXIOTAPE=The number of tape I/O EXCPS multiplied by the tape charge unit
- CXIODSK1=The number of disk type 1 I/O EXCPS multiplied by the Disk1 charge unit
- CXIODSK2=The number of disk type 2 I/O EXCPS multiplied by the Disk1 charge unit
- CXIODSKX=The number of all other disk type I/O EXCPS multiplied by the DISKX charge unit
- CXIODSKT=CXIODSK1 + CXIODSK2 + CXIODSKX (total disk I/O charge)
- CXIOOTH=All non tape or disk EXCPS multiplied by the ‘other’ charge unit
- CXIOCST=CXIODSKT + CXIOTAPE + CXIOOTH multiplied by the CPU weighting factor relative to this record

Syntax

```
%EXCHGIO    tape   disk1   disk2   diskx   other
```

`tape`

The tape EXCP I/O charge unit.

`disk1`

Disk type 1 (as defined by the DSKTYPE1 parameter of the JIFOPTS macro) EXCP I/O charge unit.

`disk2`

Disk type 2 (as defined by the DSKTYP2 parameter of the JIFOPTS macro) EXCP I/O charge unit.

`diskx`

The EXCP I/O charge unit for all other disk types.

`other`

The EXCP I/O charge unit for devices other than tape or disk drives.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
* I/O CHARGING ALGORITHM
*      ARE WE PROCESSING STEP RECORDS
IF XFLAG2 = X
   CXIOTAPE = CSIOTP  * &TAPE
   CXIODSK1 = CSIODK1 * &DISK1
   CXIODSK2 = CSIODK2 * &DISK2
   CXIODSKX = CSIODKX * &DISKX
   CXIODSKT = CXIODSK1 + CXIODSK2 + CXIODSKX
   CXIOOTH  = CSIOOTH * &OTHER
   CXIOCST  = (CXIODSKT + CXIOTAPE +CXIOOTH) * CXWGTCPU
ELSE
   CXIOTAPE = CMIOTP  * &TAPE
   CXIODSK1 = CMIODK1 * &DISK1
   CXIODSK2 = CMIODK2 * &DISK2
   CXIODSKX = CMIODKX * &DISKX
   CXIODSKT = CXIODSK1 + CXIODSK2 +CXIDSKX
   CXIOOTH  = CMIOOTH * &OTHER
   CXIOCST  = (CXIODSKT + CXIOTAPE + CXIOOTH) * CXWGTCPU
END-IF
```

Note: It is your responsibility to move an X to the field XFLAG2 prior to this routine if I/O costs are being developed from step processing.

Example

```
%EXCHGIO .14  .19  .19  .19  .10
```

In this example, tape EXCPS costs .14 charge units, all disk EXCPS costs .19 charge units, and all other EXCPS costs .10 charge units.

TPUT and TGET Charging — EXCHGTPG

The EXCHGTPG routine calculates the cost of TGETS and TPUTS per TSO session and the total cost of TGETS and TPUTS.

Syntax

```
%EXCHGTPG      tgetcost   tputcost
```

`tgetcost`

The charge unit for each TGET completed during the length of the session.

`tputcost`

The charge unit for each TPUT completed during the length of the session.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
F XFLAG1 EQ Y_FLAG_
   CXPERTGET = CMTSOTG * &TGETCOST
   CXPERTPUT = CMTSOTP * &TPUTCOST
   CXTPTGCST = CXPERTGET + CSPERTPUT
END-IF
```

Example

```
%EXCHGTPG .20  .31
```

Unit Record Device Charging — EXCHGUR

The EXCHGUR routine develops unit record usage cost, either the per line printed cost or per card punched cost. This routine yields three fields:

- CXPERLIN=The number of lines printed multiplied by the line charge unit
- CXPERCRD=The number of cards punched multiplied by the punched charge unit
- CXURCST=CXPERLIN + CXPERCRD

Syntax

```
%EXCHGUR     lines    punched
```

`lines`

Indicates the lines per line printed charge unit.

`punched`

Indicates the per card punched charge unit.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF XFLAG1 EQ Y_FLAG_
   CXPERLIN = CMLINES * &LINECOST
   CXPERCRD = CMCARDP * &CARDCOST
   CXURCST = CXPERLIN + CXPERCRD
END-IF
```

Example

```
%EXCHGUR .01  .05
```

In this example, each line printed costs .01 charge units, and each card punched costs .05 charge units.

Combined Costs — EXCOSTS

The EXCOSTS routine demonstrates the ability to build a single macro to call the two weighting and three charging routines To build a billing algorithm.

Syntax

```
%EXCOSTS     (none)
```

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
*
* SETUP WEIGHTINGS FOR PRIORITY
*
%EXWGTPTY LE 7 1
%EXWGTPTY GQ 8 2
%EXWGTPTY GQ 9 3
*
* SETPU WEIGHTINGS FOR CPU
*
%EXWGTCPU H158 1.0
%EXWGTCPU 4341 1.5
*
* CALCULATE COSTS FOR I/O
*
%EXCHGIO  .14  .19  .19  .19  .10
*
* NOW ADD TOTAL
*
%EXCHGUR .01  .05
DEBIT = CXCPUCST + CXIOCST
IF XFLAG1 EQ Y FLAG
   DEBIT = DEBIT + CXURCST
END-IF
*
* END OF EXCOSTS
```

Note: These routines are copies of the examples previously described.

One additional field is developed in this routine. DEBIT represents the total charge for the entire cost of the job.

Example Billing Routines

Seven example CA-PanAudit Plus routines (JIFBEX01 through 07) are shown on the following pages to demonstrate how billing routines can be created from the SMF data. They use the consolidated file and a billing algorithm to translate various statistical information into charges for use in billing or cost accounting.

Note: These examples are for demonstration only and are not a supported portion of the JIF utility.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEXnn     startdate   enddate
```

Where nn = the number 01 through 07.

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Table Lookup

Use is made of two tables:

- DEPTTAB–Department table
- CREDTAB–Credit/Budget table

DEPTTAB

For several of the JIF billing examples to translate the department keys specified in the job accounting information, a department table, called DEPTTAB, must be created as follows:

```
Positions
 1 -  4      Department key
 5 - 35      Department name
36 - 80      Unused
```

Example

```
//DEPTTAB   DD   *
ACCOACCOUNTING
MAILMAIL ROOM
SHIPSHIPPING & RECEIVING
...
/*
```

CREDTAB

Several of the JIF billing examples allow crediting and budgeting of cost areas by using the first 2 characters of the department field as a key into a credit/budget table. In order for these examples to use this option, the credit/budget table, called CREDTAB, must be created as follows:

```
Positions
 1 -  2     Cost area key (First 2 characters of DEPTTAB)
 3 - 11     Credit amount for Cost area
12 - 12     Blank
13 - 21     Budget amount for Cost area
22 - 80     Unused
```

Example

```
//CREDTAB   DD   *
AC001000000 001000000
MA000500000 000500000
SH001800000 001800000
...
/*
```

Invoice Ledger — JIFBEX01

The JIFBEX01 routine demonstrates the ability to create an invoice ledger report.

Syntax

```
%JIFREC      YNNNNNNN
%JIFBEX01    startdate   enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      PERFORM COLLECT_DATA
      PRINT JIFBEX01
      REPORT_FLAG_ = YES_
   END-IF
END-IF
*
%JIFPROCS JIF BEX01
*
COLLECT_DATA. PROC
SEARCH DEPTTAB WITH DEPTKEY GIVING DEPTRES
IF NOT DEPTTAB
   DERROR = 'UNKNOWN DEPT. = '
   ERRDEPT = DEPTKEY
END-IF
*
%EXCOSTS
*
END-PROC
*
```

```
REPORT JIFBEX01 SUMMARY SUMCTL (HIAR) SPACE 1 SUMSPACE 2
SEQUENCE DEPTFLD
CONTROL FINAL DEPTFLD
TITLE 01 'INVOICE LEDGER  (ROUTINE JIFBEX01)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND &ENDDATE'
HEADING DEPTFLD ('COST CENTER')
HEADING DBTCHG  ('DEBIT')
HEADING CRTAMT  ('CREDIT')
HEADING TOTCHG  ('TOTAL' 'CHARGE')
HEADING BGTAMT  ('BUDGET')
HEADING BGTDEV  ('OVER-' 'UNDER+' 'BUDGET')
HEADING BGTPCT  ('PERCENTAGE' 'OF' 'BUDGET')
LINE 01 DEPTFLD DBTCHG CRTAMT TOTCHG BGTAMT BGTDEV BGTPCT
*
REPORT-INPUT. PROC.
IF DEPTKEY NE HOLDKEY
   HOLDKEY = DEPTKEY
   SEARCH CREDTAB WITH HOLDKEY2 GIVING HOLDDESC
   IF NOT CREDTAB
      HOLDCRED = 0
      HOLDBUDG = 0
   END-IF
ELSE
   HOLDCRED = 0
   HOLDBUDG = 0
END-IF
DBTCHG = DEBIT
CRTAMT = HOLDCRED
TOTCHG = DBTCHG - CRTAMT
BGTAMT = HOLDBUDG
BGTDEV = BGTAMT - TOTCHG
SELECT
END- PROC
*
BEFORE-BREAK. PROC
IF BGTAMT EQ 0
   BGTPCT = BGTDEV * N100_
ELSE
   BGTPCT = BGTDEV * N100_/ BGTAMT
END-IF
END-PROC
```

Detail Charge Audit — JIFBEX02

The JIFBEX02 routine demonstrates the ability to create a detail charge audit report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX02     startdate   enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing code.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      IF XFLAG2 EQ Y_FLAG_
         PERFORM COLLECT_DATA
         PRINT JIFBEX02
         REPORT_FLAG_ = YES_
      END-IF
   END-IF
END-IF
*
COLLECT_DATA. PROC
*
%JIFPROCS JIF BEX01
*
%EXCOSTS
*
DISKTOT = CSDISK1 + CSDISK2 + CSDISKX
IOTOT   = CSIODK1 + CSIODK2 + CSIODKX
END-PROC
*
REPORT JIFBEX02 SPACE 1 SUMSPACE 1
SEQUENCE CMJOBNM
CONTROL FINAL
TITLE 01 'DETAIL CHARGE AUDIT (ROUTINE JIFBEX02)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND &ENDDATE'
HEADING CMJOBNM  ('JOB' 'NAME')
HEADING CSPROGN  ('PROGRAM' 'NAME')
HEADING CMJOBCL  ('C' 'L' 'A')
HEADING CMJOBPT  ('P' 'R' 'I')
HEADING CSCPU    ('STEP' 'CPU' 'TIME')
HEADING CSIOTP   ('TAPE' 'I/O')
HEADING IOTOT    ('DISK' 'I/O')
HEADING CSTAPNUM ('#' 'TP')
HEADING DISKTOT  ('#' 'DK')
HEADING CMLINES  ('LINES' 'PRINTED')
HEADING CMCARDR  ('CARDS' 'READ')
HEADING CMCARDP  ('CARDS' 'PUNCHED')
HEADING DEBIT    ('TOTAL' 'CHARGE')
LINE 01 CMJOBNM CSPROGN CMJOBCL CMJOBPT CSCPU CSIOTP IOTOT CSTAPNUM -
        DISKTOT CMLINES CMCARDR CMCARDP DEBIT
```

Job Charge Summary — JIFBEX03

The JIFBEX03 routine demonstrates the ability to create a job charge summary report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX03     startdate enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      PERFORM COLLECT_DATA
      PRINT JIFBEX03
      REPORT_FLAG_ = YES_
   END-IF
END-IF
*
%JIFPROCS JIF BEX03
*
COLLECT_DATA. PROC
SEARCH DEPTTAB WITH DEPTKEY GIVING DEPTRES
IF NOT DEPTTAB
   DERROR = 'UNKNOWN DEPT. = '
   ERRDEPT = DEPTKEY
END-IF
*

%EXCOSTSIF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      IF XFLAG2 EQ Y_FLAG_
         PERFORM COLLECT_DATA
         PRINT JIFBEX02
         REPORT_FLAG_ = YES_
      END-IF
   END-IF
END-IF
*
COLLECT_DATA. PROC
*
%JIFPROCS JIF BEX01
*
%EXCOSTS
*
DISKTOT = CSDISK1 + CSDISK2 + CSDISKX
IOTOT   = CSIODK1 + CSIODK2 + CSIODKX
END-PROC
*
REPORT JIFBEX02 SPACE 1 SUMSPACE 1
SEQUENCE CMJOBNM
CONTROL FINAL
TITLE 01 'DETAIL CHARGE AUDIT (ROUTINE JIFBEX02)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND &ENDDATE'
HEADING CMJOBNM  ('JOB' 'NAME')
HEADING CSPROGN  ('PROGRAM' 'NAME')
HEADING CMJOBCL  ('C' 'L' 'A')
```

```
HEADING CMJOBPT  ('P' 'R' 'I')
HEADING CSCPU    ('STEP' 'CPU' 'TIME')
HEADING CSIOTP   ('TAPE' 'I/O')
HEADING IOTOT    ('DISK' 'I/O')
HEADING CSTAPNUM ('#' 'TP')
HEADING DISKTOT  ('#' 'DK')
HEADING CMLINES  ('LINES' 'PRINTED')
HEADING CMCARDR  ('CARDS' 'READ')
HEADING CMCARDP  ('CARDS' 'PUNCHED')
HEADING DEBIT    ('TOTAL' 'CHARGE')
LINE 01 CMJOBNM CSPROGN CMJOBCL CMJOBPT CSCPU CSIOTP IOTOT CSTAPNUM -
        DISKTOT CMLINES CMCARDR CMCARDP DEBIT

*
END-PROC
*
REPORT JIFBEX03 SPACE 1 SUMSPACE 2 DTLCTL EVERY SUMCTL NONE
SEQUENCE DEPTFLD
CONTROL FINAL DEPTFLD
TITLE 01 'JOB CHARGE SUMMARY (ROUTINE JIFBEX03)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND ENDDATE'
HEADING DEPTFLD  ('COST CENTER')
HEADING CMJOBNM  ('JOBNAME')
HEADING TALLY    ('NUMBER' 'OF' 'JOBS')
HEADING CXCPUCST ('PROCESS' 'CHARGES')
HEADING CXIOCST  ('I/O' 'CHARGES')
HEADING CXURCST  ('U/R' 'CHARGES')
HEADING DEBIT    ('TOTAL' 'CHARGES')
LINE 01 DEPTFLD CMJOBNM TALLY CXCPUCST CXIOCST CXURCST DEBIT
```

Data Center Cost Recovery — JIFBEX04

The JIFBEX04 routine demonstrates the ability to create a data center cost recovery report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX04     startdate   enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      PERFORM COLLECT_DATA
      PRINT JIFBEX04
      REPORT_FLAG_ = YES_
   END-IF
END-IF
*
%JIFPROCS JIF BEX04
*
COLLECT_DATA. PROC
SEARCH DEPTTAB WITH DEPTKEY GIVING DEPTRES
IF NOT DETPTAB
   DERROR = 'UNKNOWN DEPT. = '
   ERRDEPT = DEPTKEY
END-IF
*
%EXCOSTS
*
END-PROC
*
REPORT JIFBEX04 SPACE 1 SUMSPACE 2 DTLCTL EVERY SUMCTL NONE
SEQUENCE DEPTFLD
CONTROL FINAL DEPTFLD
TITLE 01 'DATA CENTER COST RECOVERY (ROUTINE JIFBEX04)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND &ENDDATE'
HEADING DEPTFLD  ('COST CENTER')
HEADING TALLY    ('NUMBER' 'OF' 'JOBS')
HEADING CXCPUCST ('PROCESS' 'CHARGES')
HEADING CXIOCST  ('I/O' 'CHARGES')
HEADING CXURCST  ('U/R' 'CHARGES')
HEADING DEBIT    ('TOTAL' 'CHARGES')
LINE 01 DEPTFLD TALLY CXCPUCST CXIOCST CXURCST DEBIT
```

Monthly Utilization by Cost Center — JIFBEX05

The JIFBEX05 routine demonstrates the ability to create a monthly utilization by cost center report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX05     startdate enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      PERFORM COLLECT_DATA
      PRINT JIFBEX05
      REPORT_FLAG_ = YES_
   END-IF
END-IF
*
%JIFPROCS JIF BEX05
*
COLLECT_DATA. PROC
SEARCH DEPTTAB WITH DEPTKEY GIVING DEPTRES
IF NOT DEPTTAB
   DERROR = 'UNKNOWN DEPT. = '
   ERRDEPT = DEPTKEY
END-IF
*
%EXCOSTS
*
IOTOT = CMIOTP + CMIODK1 + CMIODK2 + CMIODKX + CMIOOTH
MONTKEY = DATE
SEARCH MONTHS WITH MONTKEY GIVING MONTFLD
END-PROC
*
REPORT JIFBEX05 SPACE 1 SUMSPACE 2 SUMMARY
SEQUENCE DEPTFLD CMCOUNT2
CONTROL FINAL DEPTFLD MONTFLD
TITLE 01 'MONTHLY UTILIZATION SUMMARY BY COST CENTER (ROUTINE
JIFBEX05)'
TITLE 02 'DATA SELECTED BETWEEN DATES  &STARTDATE AND &ENDDATE'
TITLE 03 'YEAR = 19' OVRDATE
HEADING DEPTFLD  ('COST CENTER')
HEADING MONTFLD  ('MONTH')
HEADING TALLY    ('NUMBER' 'OF' 'JOBS')
HEADING CMTMELAP ('ELAPSED' 'TIME')
HEADING CMTMCPU  ('CPU' 'TIME')
HEADING IOTOT    ('TOTAL' 'I/O' 'COUNT')
HEADING DEBIT    ('TOTAL' 'CHARGES')
LINE 01 DEPTFLD MONTFLD TALLY CMTMELAP CMTMCPU IOTOT DEBIT
```

Data Processing Invoice — JIFBEX06

The JIFBEX06 routine demonstrates the ability to create a data processing invoice report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX06     startdate enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-PanAudit Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
      PERFORM COLLECT_DATA
      PRINT JIFBEX06
      REPORT_FLAG_ = YES_
   END-IF
END-IF
*
%JIFPROCS JIF BEX06
*
COLLECT_DATA. PROC
SEARCH DEPTTAB WITH DEPTKEY GIVING DEPTRES
IF NOT DEPTTAB
   DERROR = 'UNKNOWN DEPT. = '
   ERRDEPT = DEPTKEY
END-IF
*
%EXCOSTS
*
END-PROC
*
REPORT JIFBEX06 SPACE 1 SUMSPACE 2 SUMMARY
SEQUENCE DEPTFLD
CONTROL FINAL DEPTFLD
TITLE 01 'DATA PROCESSING INVOICE (ROUTINE JIFBEX06)'
TITLE 02 'DATA SELECTED BETWEEN DATES - &STARTDATE AND &ENDDATE'
HEADING DEPTFLD ('COST CENTER')
HEADING DBTCHG  ('DEBIT')
HEADING CRTAMT  ('CREDIT')
HEADING TOTCHG  ('TOTAL' 'CHARGE')
LINE 01 DEPTFLD DBTCHG CRTAMT TOTCHG
*
REPORT-INPUT. PROC
IF DEPTKEY NE HOLDKEY
   HOLDKEY = DEPTKEY
   SEARCH CREDTAB WITH HOLDKEY2 GIVING HOLDDESC
   IF NOT CREDTAB
      HOLDCRED = 0
   END-IF
ELSE
   HOLDCRED = 0
END-IF
DBTCHG = DEBIT
CRTAMT = HOLDCRED
SELECT
END-PROC
```

```
*
BEFORE-BREAK. PROC
TOTCHG + DTBCHG CRTAMT
END-PROC
*
```

TSO User Charging Report — JIFBEX07

The JIFBEX07 routine demonstrates the ability to create a TSO user charging report.

Syntax

```
%JIFREC       YNNNNNNN
%JIFBEX07     startdate   enddate
```

`startdate`

The date (YYDDD) that selection of job information is to begin.

`enddate`

The date (YYDDD) that selection of job information is to end.

Routine Code

The following CA-Easytrieve Plus routine is intended as an example to demonstrate how JIF creates a job accounting/billing system using CA-PanAudit Plus. As such, it is not part of the supported product, but serves as a model to assist you in developing job accounting/billing coding.

```
IF CMFLAG1 EQ C_FLAG_
   IF CMFLAG2 EQ T_FLAG_
      IF CMCOUNT2 = &STARTDATE THRU &ENDDATE
         PERFORM COLLECT_DATA
         PRINT JIFBEX07
         REPORT_FLAG_ = YES_
      END-IF
   END-IF
END-IF
*
%JIFPROCS JIF BEX07
*
COLLECT_DATA. PROC
*
* SETUP WEIGHTINGS FOR PRIORITY
*
%EXWGTPTY LE 7 1
%EXWGTPTY GQ 8 2
%EXWGTPTY GQ 9 3
*
* SETUP WEIGHTINGS FOR CPU
*
%EXWGTCPU H158 1.0
%EXWGTCPU 4341 1.5
```

```
*
* CALCULATE COSTS FOR CPU UTILIZATION
*
%EXCHGCPU 7.45 9.90
*
* CALCULATE COSTS FOR I/O
%EXCHGIO  .14  .19  .19  .19  .10
*
* CALCULATE COSTS FOR UNIT RECORD DEVICES
*
%EXCHGUR   .01  .05
*
* CALCULATE COSTS FOR TGETS AND TPUTS
*
%EXCHGTPG .05  .01
*
* CALCULATE TOTALS AND REPORT SPECIFIC CHARGES
*
CXCONCHG = CXWGTCPU * CMTMELAP * 1.00
DEBIT = CXCONCHG
DEBIT = DEBIT + CXCPUCST
DEBIT = DEBIT + CXIOCST
DEBIT = DEBIT + CXURCST
DEBIT = DEBIT + CXTPTGCST
CXTOTIO = CXIOCST + CXURCST
*
END-PROC
*
REPORT JIFBEX07 SPACE 1 SUMSPACE 2 DTLCTL EVERY SUMCTL NONE
SEQUENCE CMJOBNM CMCOUNT2 CMJOBNO
CONTROL FINAL CMJOBNM
TITLE 01 'USER CHARGING REPORT (ROUTINE JIFBEX07)'
TITLE 02 'DATA SELECTED BETWEEN DATES  &STARTDATE AND &ENDDATE'
HEADING CMJOBNM   ('USER' 'ID')
HEADING CMDTSTT   ('LOGON' 'DATE')
HEADING CMJOBNO   ('TSU' 'NUMBER')
HEADING CXCONCHG  ('CONNECT' 'CHARGE')
HEADING CXCPUCST  ('PROCESS' 'CHARGE')
HEADING CXTOTIO   (I/O' 'CHARGE')
HEADING CXTPTGCST ('TGET/TPUT' 'CHARGE')
HEADING DEBIT     ('TOTAL' 'CHARGE')
LINE 01 CMJOBNM CMDTSTT CMJOBNO CXCONCHG +
        CXCPUCST CXTOTIO DEBIT
```

Chapter

11 Graphing Facility

This chapter describes the CA-PanAudit Plus graphing facility.

Using the Facility

The CA-PanAudit Plus graphing facility lets you produce graphic representation of data in your files. A graph is useful in determining relationships or trends in data and as a visual aid for data presentations in reports, analyses, and briefings.

The graphing facility produces four types of graphs:

- Standard bar graphs
- Histograms
- Plot graphs
- Deviation bar graphs

The graphing facility:

- Is designed primarily for use as a stand-alone program, graphing values from an input file
- Uses any file that can be input to CA-PanAudit Plus
- Operates by means of the macro invocation of CA-PanAudit Plus statements
- Uses freeform, nonpositional English-like keywords to control the graph format
- Produces all graphs on a standard line printer, eliminating the need for specialized output equipment

Coding

The coding required to execute the graphing facility contains three essential parts:

- The Library section that describes your input file
- The FILE statement that describes the Keyword File

Statements that invoke the graphing facility

The required statements are discussed in the following two topics. Sample JCL is illustrated later in this chapter.

Restrictions

You cannot use the Graphing facility with other CA-PanAudit Plus routines. You can use user-defined logic (for example, screening of input data) only if certain rules are observed. These rules are fully explained later in Required Statements later in this chapter.

The subroutines used by the Graphing facility are not reentrant. This means that multiple graphing jobs cannot be used in the same JCL job step. You must run each job individually.

Minimum and maximum values permitted as input to any of the graph routines are:

```
Minimum:   -21,474,836.48
Maximum:   +21,474,836.47
```

If you input a value outside this range, unpredictable results occur.

The Graphing facility requires 450 KB of storage for execution. Also, for DOS execution, the EXITSTR parameter in the environment section of a CA-Easytrieve Plus program must be specified as follows:

```
PARM EXITSTR 180
```

See the DOS JCL example at the end of this chapter. See the CA-Easytrieve Plus *Reference Guide* for an explanation of the EXITSTR parameter.

The graphing routines cannot specify a database file as the input file.

Required Statements

The elements required to execute the graphing facility consist of the following major parts:

- FILE statements:
  - Input FILE statement
  - Keyword FILE statement
- Graphing facility invocation statement
- An END card
- File containing keyword commands

FILE Statements

The two kinds of FILE statements are input and keyword.

Input FILE Statement

The graphing facility accepts as input any file that can be read by CA-PanAudit Plus. The Input FILE statement is used to define this file.

Keyword FILE Statement

Keywords are the commands you use to create customized graphic output. They are English-like terms (such as COLS, WIDTH, and ROWS) that specify the physical form of the graphs. You code these keywords in a file that you define and place within the job stream. The keywords to be coded are described individually for each graph type.

You define the keyword file by a FILE statement coded with the CARD parameter. The file name for the keyword file can be any combination of characters valid to CA-PanAudit Plus.

Graphing Facility Invocation Statement

The generalized syntax for the graphing facility invocation statement follows.

Syntax

```
%GRAPH1 infile type parmfile parms control-field f1 f2 f3 f4 f5 f6 sortfield
%GRAPH2
```

`infile`

The name of the file that supplies input to GRAPH1 activity. A valid name can be any previously defined file.

`type`

The type of graph to be produced. Valid values are BAR, HIST, PLOT, or DEV.

`parmfile`

The name of the input file that contains the graphing facility keyword parameters. See Operation (following) for additional information about parmfile.

`parms`

The 80-byte field in the keyword file that contains the keyword statements. In the example shown in parmfile, the value for parms is PRM.

`control-field`

The name of the nonquantitative alphanumeric field that contains the control break information for each graph (such as REGION or DEPT). You can name only one control field for each execution of the graphing facility.

`f1, f2, f3, f4, f5, f6`

The fields f1…f6 are used as input to the various graphing facility routines. Each routine uses these fields differently. All six parameters must be represented in the %GRAPH1 statement by an alphanumeric string separated by blanks.

Note: Any unused parameters must be coded with a zero.

`sortfield`

The name of the field used for sorting the input file. This parameter has restrictions which are described for each graph type.

`%GRAPH2`

Ending statement for the %GRAPH1 activity.

Operations

Parmfile is defined using the CARD parameter in the FILE Statement as shown in the CA-Easytrieve Plus *Reference Guide*. The format is:

```
FILE INCRD CARD
PRM  1  80  A
```

User Defined Screening Code

If records from the input file will be bypassed or otherwise altered before being input to the graphing facility, put the associated CA-Easytrieve Plus logic between the two graphing invocation statements (%GRAPH1 and %GRAPH2). If this code is placed anywhere else, syntax errors are generated. Input data screening for the graphing facility is identical to screening procedures for other routines.

Example

The following generalized example illustrates the sequence in which the required statements must appear in the graphing facility coding. Complete JCL is provided later in this chapter.

The graphing invocation statements, introduced by %GRAPH1, must be followed by an ending statement, %GRAPH2, followed in turn by and END statement and the file containing the keywords.

```
                      FILE MYFILE
Input FILE            SALESYTD 1 8 N 2
Statement             AMTDUE 32 8 N 2
                      REGION 49 4 A

Keyword FILE          FILE INCRD CARD
Statement             PRM  1  80  A

Graphing              %GRAPH1 MYFILE BAR INCRD PRM REGION SALESYTD
Invocation                    AMTDUE 0 0 0 0 REGION
Statements            %GRAPH2

END Card              END

                      1TITLE SAMPLE TITLE LINE FOR BAR GRAPH
                      LINE 150000.00
File                  FLD1,VAR,TOT
Containing            FLD2,VAR,TOT
Keyword               BARKEY1 SALESYTD TOT
Commands              BARKEY2 AMTDUE TOT
for Specific          ROWS 40
Graphs                COLS 99
                      WIDTH 5
                      SPACE 2
                      BARS 1
                      1FOOT TEST FOOTER FOR BAR GRAPH
```

Bar Graph

The standard bar graph facility produces a multiple-bar graphic that can use up to six variables (f1 through f6). The variables can be specified as fields from the input file, or entered as a constant, such as Sales-Target and Year-to-Date.

The same field can be used more than once on the same bar graph. This allows you to graph multiple values (minimum, maximum, mean, and so on) of a specific field, using one graph for all values instead of one graph for each value.

Syntax

```
%GRAPH1 infile BAR parmfile parms control-field f1 f2 f3 f4 f5 f6 sortfield
%GRAPH2
```

`fn`

You can use all six of the f1…f6 parameters. Each parameter used must have an associated FLDn keyword (see the following) in your Keyword File. The BARS keywords must indicate the number of f1…f6 parameters being used. This results in one bar being graphed for each f1…f6 parameter specified. Where desired, quoted literals can be used in place of field names. Code all unused parameters with zeroes.

`sortfield/control-field`

Sortfield must be the same field as control-field.

Operation

Values for each variable entered in the routine are accumulated until the value of the control field changes. At that time a control break occurs, and a group of bars is produced, one for each variable. This process is repeated for each value of the control field until an end-of-file condition is reached on the input file.

Each bar on the printed output consists of a different special character that is supplied by the routine. This allows you to differentiate among variables. A legend can be printed at the top of the graph identifying the variable represented by each character or bar.

Multiple graphs are produced when the output exceeds the space of one graph. When this occurs, the graph continues on another output page.

Keywords

The following keywords are used to format the bar graph. Coding examples follow the keyword explanations.

`1TITLE`

`2TITLE`

`3TITLE`

FORMAT—
```
1TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
3TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

FUNCTION—Three optional nTITLE lines are allowed, each of which can contain up to 40 characters of titling information. Any or all lines can be omitted. The nTITLE line sequence numbers are required. Each nTITLE line results in one title line being printed at the top of the graph.

DEFAULT—Output report title lines are blank if no titling information is entered.

`ROWS`

FORMAT— `ROWS nn`

FUNCTION— Specifies the number of horizontal rows to be used per graph.

MINIMUM— 1

MAXIMUM— 40

DEFAULT— 40

`COLS`

FORMAT— `COLS nn`

FUNCTION— Specify the number of columns to be used as the total width of the graph. If the number is less than the total combined width of the graph desired (based on the number and width of the bars requested), the system uses the output default value. COLS nn includes both bars and spaces.

MINIMUM— 32

MAXIMUM— 99

DEFAULT— 32

`LINE`

FORMAT— `LINE value M`

FUNCTION— When you specify the LINE parameter, the bar graph prints an additional value that is represented on the graph by a broken double line printed horizontally across the output report. This line is equivalent to the value's placement within the graph and results in a clear, visual comparison of the LINE value with the other data being graphed. A valid value for the LINE parameter is an actual numeric value between –2,147,483,648 and +2,147,483,647.

For example, LINE 500000.00 results in a significant value of 500,000.00 being displayed as a line across the graph. It is not necessary to code a decimal point when using whole numbers as significant values. (This could have been coded LINE 500000.)

To produce a mean (or average) line of the values graphed, code an M after the LINE keyword. This automatically calculates and prints the mean value of the input file.

If the value of the significance line is out of range of the file being graphed, the line appears on a level with the highest value reported.

DEFAULT— No line shown.

1FOOT

2FOOT

FORMAT— `1FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
`2FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

FUNCTION— You can specify two optional nFOOT lines, each of which can contain up to 40 characters of alphanumeric information. These lines appear on the last page of the output and are printed under the graph. The nFOOT sequence number must be given.

DEFAULT— Blank lines

WIDTH

FORMAT—`WIDTH nn`

FUNCTION—Specify the width, in characters, of each bar.

MINIMUM—1

MAXIMUM—Limited only by the number of bars desired, the space between each bar, and the width of the final printed output. If a bar's width causes it to compromise any of these factors, the default value of 5 is used. There is no maximum.

DEFAULT—5

If the total width of the graph (using the COLS keyword) is less than the total combined width of all bars, the default value of 5 is chosen by the routine. This allows a maximum of six bars to fit within the minimum 32 columns available for reporting.

SPACE

FORMAT—`SPACE nn`

FUNCTION—Specify the number of blank spaces to appear between each bar group at control breaks.

MINIMUM—1

MAXIMUM—Limited only by the number of bars desired, the width of each bar, and the width of the final printed output. If the number of spaces requested causes it to compromise any of these factors, the default value of 1 is used by the routine. There is no maximum value for this keyword.

DEFAULT—1

BARS

FORMAT—`BARS n`

FUNCTION— Establishes the number of bars to be graphed in each control group. The value you code must correspond to the number of nonzero Fn parameters specified in %GRAPH1.

MINIMUM—1

MAXIMUM—6 (If more than six bars are requested, BARSdefaults to 1.)

DEFAULT—None. This keyword is required; the routine will not execute if this parameter is omitted.

`FLDn,PARM1,PARM2`

FORMAT—`FLDn,parm1,parm2`

(where n is the sequence number of the bar being graphed). Commas are required between FLDn and parm1, and between parm1 and parm2.

FUNCTION—Determines the method used to compute output values. By using different parameters you can control whether each bar represents a constant value or a variable, and in the case of a variable, whether the mean, minimum, maximum, count, or total value of the variable is graphed.

`parm1`—This field is set either to CON (CONstant) or VAR (VARiable).

The CON option, which does not have an associated parm2 value, results in the named field being used as a constant for reporting purposes. It is recommended that you place the constant value in the invocation of GRAPH1. (See Example One). Because no arithmetic calculations are performed on this field, the value of the field, at a control break, is used for the final graphic output.

The VAR option is used with one of the following parm2 options. All values resulting from the use of the VAR option are taken at the control break and are reported for each control field.

parm2

TOT—Causes the bar graph routine to calculate and display the TOTal amount of the specified field for each control break.

MEAN—This causes the routine to calculate and display the MEAN (or average) value of all input records within each control break.

MAX—Graphs the MAXimum value found on the file within each control break.

MIN—Graphs the MINimum value found on the file within each control break.

CNT—Calculates and displays the number of occurrences of a field within control breaks. Generally, CNT is used with only one field (bar), as it gives a total of the number of records occurring within each control break.

For example, if there are 51 records for Region 101, with REGION being the control field, CNT will show a frequency of 51.

DEFAULT—None.

CONKEY

FORMAT—CONKEY xxxxxxxxxxxx

FUNCTION—This keyword can contain up to 12 alphanumeric characters. It is used as an informational field by which you describe the control field being used (for example, REGION1).

DEFAULT—When CONKEY is omitted, the control value field on the output is blank.

BARKEYn

FORMAT—BARKEYn xxxxxxxxxxxx

(where n is the sequence number, from 1 to 6, of the bar being graphed).

FUNCTION—This keyword can contain up to 12 alphanumeric characters. It is used as a descriptor field, allowing you to explain the keys, or special characters, used to display each bar. This field is displayed as a legend on the printed output. You can specify from one to six BARKEYn keywords.

DEFAULT—When BARKEYn is omitted, the legend on the printed output is blank.

Examples

The following are two examples of the bar graph. Each example consists of the input, the bar graph produced, and an explanation of the graph.

Example One

Input

```
FILE INFILE ...
REGION    1    2    N
SD        3    8    P  2
FILE INCRD CARD
PRM       1   80    A
...
%GRAPH1 INFILE BAR INCRD PRM REGION SD SD SD SD 300000 0 REGION
%GRAPH2
END
```

Output

```
            SAMPLE FOR BAR GRAPH EXAMPLE 1
        ALL VALUES ARE FOR SALES-YEAR-TO-DATE
          SHOWING MIN, MAX, MEAN, TOT AND CONSTANT

               * REPRESENTS SALESYTD MIN
               # REPRESENTS SALESYTD MAX
               X REPRESENTS SALESYTDMEAN
               + REPRESENTS SALESYTD TOT
               @ REPRESENTS $300,000.00

               ALL VALUES HAVE BEEN SCALED BY A FACTOR OF        100
               CONTROL VALUE: REGION
        +------------------------------------------------------------*
3999.75 I +++                                                        *
3861.83 I +++                                                        *
3723.90 I +++ +++                                                    *
3585.98 I              +++ +++                                       *
3448.06 I              +++ +++                                       *
3310.14 I          +++ +++ +++                                       *
3172.21 I          +++ +++ +++                                       *
3034.29 I          +++ +++ +++                                       *
2896.37 I          +++@@@           +++@@@ +++@@@                    *
2758.45 I          +++@@@           +++@@@ +++@@@                    *
2620.52 I==========+++@@@===========+++@@@===========+++@@@==========*
2482.60 I          +++@@@           +++@@@ +++@@@                    *
2344.68 I          +++@@@           +++@@@ +++@@@                    *
2206.76 I          +++@@@           +++@@@           +++@@@          *
2068.83 I          +++@@@           +++@@@           +++@@@          *
1930.91 I          +++@@@           +++@@@           +++@@@          *
1792.99 I          +++@@@           +++@@@           +++@@@          *
1655.07 I          +++@@@           +++@@@              @@@          *
1517.14 I          +++@@@           +++@@@           +++@@@          *
1379.22 I          +++@@@           +++@@@           +++@@@          *
1241.30 I          +++@@@           +++@@@           +++@@@          *
1103.37 I          +++@@@           +++@@@           +++@@@          *
 965.45 I          +++@@@           +++@@@           +++@@@          *
 827.53 I          +++@@@     ###   +++@@@     ###   +++@@@          *
 689.61 I    ###   +++@@@     ###   +++@@@     ###   +++@@@          *
 551.68 I    ###XXX+++@@@     ###XXX+++@@@     ###XXX+++@@@          *
 413.76 I    ###XXX+++@@@     ###XXX+++@@@     ###XXX+++@@@          *
 275.84 I ***###XXX+++@@@     ###XXX+++@@@  ***###XXX+++@@@          *
 137.92 I ***###XXX+++@@@  ***###XXX+++@@@  ***###XXX+++@@@          *
  -0.01 I ***###XXX+++@@@  ***###XXX+++@@@  ***###XXX+++@@@          *
        +------------------------------------------------------------*
          01               02               03
  SIGNIFICANCE LINE VALUE =    2500.00 SCALED BY A FACTOR OF:    100
                         SALES YEAR-TO-DATE
                            IS BY REGION
```

Each set of bar graphs on the output is separated by two spaces (the SPACE 2 keyword). The control field chosen is REGION, which is represented as 01, 02, and 03 on the graph.

To keep the graph as simple as possible, quantitative values are factored, so values printed along the y axis do not become too large. The previous example is factored by 100. This function is performed automatically, and the factor varies depending on the size of the input values.

The increment value of the y axis is determined automatically by dividing the largest value to be graphed by the number of ROWS requested minus one. This creates a y axis with the specified number of rows ranging from 0 to the largest graphed value.

The significance line is printed at the value 2,500.00 (factored by 100) as specified by LINE 250000.00. Because no specific interval of 2,500.00 exists on the graph, the line is printed at the interval immediately above the true value of 2,500.00. The significance line in the bar graph does not overwrite the data.

On the box that outlines the graph, the right side consists of a column of asterisks (*). This indicates that the graph shown is continued to another page of output. The graph on the next page of output is not shown but would contain a row of asterisks in the left-hand column to indicate that it is a continuation of a previous page of output. This continues until the final page is reached. The right side of the graph will consist of the character I. The Bar Graph in Example Two shows a graph indicating that it is the last page of the report.

Example Two

Input to Example Two is shown next, followed by its output and an explanation of the graph.

Input

```
FILE INFILE
REGION    1   2   N
SALESYTD  3   8   P   2
FILE INCRD CARD
PRM 1 80 A
%GRAPH1 INFILE BAR INCRD PRM REGION SALESYTD 0 0 0 0 0 REGION
%GRAPH2
END
1TITLE         SAMPLE FOR BAR GRAPH EXAMPLE 2
2TITLE      DEMONSTRATES USE OF THE CNT KEYWORD
3TITLE         (NUMBER OF RECORDS PER REGION)
LINE M
FLD1,VAR,CNT
BARKEY1 RECS PER REG
CONKEY REGION
ROWS 30
COLS 50
WIDTH 5
SPACE 4
BARS 1
```

Output

```
                    SAMPLE FOR BAR GRAPH EXAMPLE 2
                 DEMONSTRATES USE OF THE CNT KEYWORD
                    (NUMBER OF RECORDS PER REGION)

                      * REPRESENTS RECS PER REG
                      CONTROL VALUE: REGION
         +---------------------------------------------------+
  85.00 I                                     *****          I
  82.07 I                                     *****          I
  79.14 I          *****              *****   *****          I
  76.21 I          *****              *****   *****          I
  73.28 I==========*****==============*****===*****==========I
  70.34 I *****    *****              *****   *****          I
  67.41 I *****    *****              *****   *****          I
  64.48 I *****    *****     *****    *****   *****          I
  61.55 I *****    *****     *****    *****   *****          I
  58.62 I *****    *****     *****    *****   *****          I
  55.69 I *****    *****     *****    *****   *****          I
  52.76 I *****    *****     *****    *****   *****          I
  49.83 I *****    *****     *****    *****   *****          I
  46.90 I *****    *****     *****    *****   *****          I
  43.97 I *****    *****     *****    *****   *****          I
  41.03 I *****    *****     *****    *****   *****          I
  38.10 I *****    *****     *****    *****   *****          I
  35.17 I *****    *****     *****    *****   *****          I
  32.24 I *****    *****     *****    *****   *****          I
  29.31 I *****    *****     *****    *****   *****          I
  26.38 I *****    *****     *****    *****   *****          I
  23.45 I *****    *****     *****    *****   *****          I
  20.52 I *****    *****     *****    *****   *****          I
  17.59 I *****    *****     *****    *****   *****          I
  14.65 I *****    *****     *****    *****   *****          I
  11.72 I *****    *****     *****    *****   *****          I
   8.79 I *****    *****     *****    *****   *****          I
   5.86 I *****    *****     *****    *****   *****          I
   2.93 I *****    *****     *****    *****   *****          I
  -0.00 I *****    *****     *****    *****   *****          I
         +---------------------------------------------------+
           01       02        03       04       05
 SIGNIFICANCE LINE VALUE =      70.83 SCALED BY A FACTOR OF: 100
```

Because only one bar was specified (BARS 1), each bar on the graph represents one control break (in this case region). The significance line (70.83) is the average of the values (LINE M). No specific interval of 70.83 exists, so this line is placed at the value immediately above the true value of 70.83.

The values on the y axis are not scaled because the largest value is 85.00, so the scaling factor is 1, and the values printed on the y axis are the actual values. The right-hand line defining the graph box consists of the character I, indicating the end of the graph output.

Histogram

The histogram produces a graph of the distribution of values on a y axis within intervals of x. The distribution shown can represent total, mean value, or frequency.

The x values are subdivided into intervals represented by upper and lower ranges. The value of y within each range of x values is represented by a bar on the output graph.

Syntax

```
%GRAPH1 infile HIST parmfile parms control-field f1 f2 f3 f4 f5 f6 sortfield
%GRAPH2
```

`fn`

The histogram uses only the f1 and f2 parameters. You must code all others with a zero. The f1 parameter gives the value of x, the f2 parameter the value of y.

`sortfield`

The sort field must be the same as the field you specify for the x value.

Operation

The f1 parameter defines the x value desired, and the f2 parameter defines the y value. You cannot use literals with the histogram routine.

Keywords

The following keywords are used to format the histogram. A coding example follows the keyword explanations.

`1TITLE`

`2TITLE`

`3TITLE`

FORMAT—
```
1TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
3TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

FUNCTION—Three optional nTITLE lines are allowed, each of which can contain up to 40 characters of titling information. Any or all lines can be omitted. The nTITLE line sequence numbers are required. Each nTITLE line will result in one title line being printed on the graph.

DEFAULT—Output report title lines are blank if no titling information is entered.

`ROWS`

FORMAT—`ROWS nn`

FUNCTION—Specify the number of horizontal rows to be used per graph.

MINIMUM—1

MAXIMUM—40

DEFAULT—40

`COLS`

FORMAT—`COLS nn`

FUNCTION—Specify the number of columns to be used as the total width of the graph. If the number is less than the total combined width of the graph desired (based on the number and width of the bars requested), the system uses the output default value.

MINIMUM—32

MAXIMUM—99

DEFAULT—32

`LINE`

FORMAT—`LINE value`

FUNCTION—When you specify the LINE parameter, the Bar Graph prints an additional value that is represented on the graph by a broken double line printed horizontally across the output report. This line is equivalent to the value's placement within the graph and results in a clear, visual comparison of the LINE value with the other data being graphed. A valid value for the LINE parameter is an actual numeric value between –2,147,483,648 and +2,147,483,647.

For example, LINE 150000.00 results in a significant value of 150,000.00 displayed as a line across the graph. It is not necessary to code a decimal point when using whole numbers as significant values. (This could have been coded LINE 150000.)

To produce a mean (or average) line of the values graphed, code an M after the LINE keyword. This automatically calculates and prints the mean value of the input file.

If the value of the significance line is out of range of the file being graphed, the line appears on a level with the highest value reported.

DEFAULT—No line shown

`1FOOT`

`2FOOT`

FORMAT—`1FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
`2FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

FUNCTION—You can specify two optional nFOOT lines, each of which can contain up to 40 characters of alphanumeric information. These lines appear on the last page of output and are printed under the graph. The nFOOT sequence number must be given.

DEFAULT—Blank lines

`WIDTH`

FORMAT—`WIDTH nn`

FUNCTION—Specify the width in characters of each bar.

MINIMUM—1

MAXIMUM—10

DEFAULT—5

Note: If the total width of the graph desired (using the COLS keyword) is less than the total combined width of all bars, the default of 5 is used by the routine. This allows a maximum of six bars to fit within the minimum 32 columns available for reporting.

`SPACE`

FORMAT—`SPACE nn`

FUNCTION—Specify the number of blank spaces to appear between each bar group at control breaks.

MINIMUM—1

MAXIMUM—10

DEFAULT—1

`XDEF`

`YDEF`

FORMAT—`XDEF xxxxxxxxxxxx`
`YDEF xxxxxxxxxxxx`

FUNCTION—These keywords can contain up to 12 alphanumeric characters each. They act as descriptor fields you can use to explain which values or fields from a file are used as the x and y values, respectively.

DEFAULT—When these keywords are omitted, the XDEF and YDEF fields on the output are blank.

`HISTYP`

FORMAT—`HISTYP,parm1`

Note: A comma is required between the keyword HISTYP and the associated parm1 value. For example, HISTYP,MEAN.

FUNCTION—Controls the type of histogram printed. By specifying one of the parm1 values, you can determine whether the total, mean, or count of y is represented for each interval.

> **TOT**—This parameter causes the routine to calculate and display the total y value within an x interval.
>
> **MEAN**—This parameter causes the routine to calculate the mean of y within a given interval.
>
> **CNT**—Gives a frequency of y within each interval.

DEFAULT—Parm1 defaults to CNT.

`INTVLS`

FORMAT—`INTVLS nn`

FUNCTION—Determines the number of intervals reported for each histogram. The intervals are determined using the routine by dividing the range of the x values in the file by the value specified for INTVLS.

MINIMUM—1

MAXIMUM—1000

DEFAULT—10

Example

The following exhibits show an example of the histogram.

Input

```
FILE INFILE
DEPT 1 2 N
RATE  26 4 N  2
LOANAMT 31 14 N  2
FILE INCRD CARD
PRM 1 80 A
%GRAPH1 INFILE HIST INCRD PRM DEPT LOANAMT RATE  0 0 0 0 LOANAMT
%GRAPH2
END
1TITLE        SAMPLE FOR HISTOGRAM GRAPH
2TITLE      GRAPH OF NUMBER OF LOANS WITHIN
3TITLE      INTERVAL DEFINED BY LOAN AMOUNT
LINE M
ROWS 35
COLS 71
WIDTH 5
SPACE 5
XDEF RATE
YDEF LOANAMT
```

```
HISTYP,CNT
INTVLS 7
```

Output

```
                          SAMPLE FOR HISTOGRAM GRAPH
                       GRAPH OF NUMBER OF LOANS WITHIN
                       INTERVAL DEFINED BY LOAN AMOUNT

             F1 VARIABLE DESCRIPTION: LOANAMT
             F2 VARIABLE DESCRIPTION: RATE
       +--------------------------------------------------------------------+
296.00 I          *****                                                     I
287.51 I          *****                                                     I
279.03 I          *****                                                     I
270.54 I          *****                                                     I
262.06 I          *****                                                     I
253.57 I          *****                                                     I
245.09 I          *****     *****                                           I
236.60 I          *****     *****                                           I
228.11 I          *****     *****                                           I
219.63 I          *****     *****                                           I
211.14 I          *****     *****                                           I
202.66 I          *****     *****                                           I
194.17 I          *****     *****                                           I
185.69 I*****     *****     *****     *****                                 I
177.20 I*****     *****     *****     *****                                 I
168.71 I*****     *****     *****     *****                                 I
160.23 I*****     *****     *****     *****                                 I
151.74 I*****=====*****=====*****=====*****=================================I
143.26 I*****     *****     *****     *****                                 I
134.77 I*****     *****     *****     *****                                 I
126.29 I*****     *****     *****     *****                                 I
117.80 I*****     *****     *****     *****                                 I
109.31 I*****     *****     *****     *****                                 I
100.83 I*****     *****     *****     *****                                 I
 92.34 I*****     *****     *****     *****                                 I
 83.86 I*****     *****     *****     *****                                 I
 75.37 I*****     *****     *****     *****                                 I
 66.89 I*****     *****     *****     *****     *****     *****             I
 58.40 I*****     *****     *****     *****     *****     *****             I
 49.91 I*****     *****     *****     *****     *****     *****             I
 41.43 I*****     *****     *****     *****     *****     *****     *****   I
 32.94 I*****     *****     *****     *****     *****     *****     *****   I
 24.46 I*****     *****     *****     *****     *****     *****     *****   I
 15.97 I*****     *****     *****     *****     *****     *****     *****   I
  7.49 I*****     *****     *****     *****     *****     *****     *****   I
       +--------------------------------------------------------------------+
         2203.00  14727.72  27252.43  39777.14  52301.85  64826.56  77351.25
           -         -        -          -         -        -           -
        14727.71  27252.42  39777.13  52301.84  64826.55  77351.25  89875.94
```

This example demonstrates the HISTYP,CNT option of the histogram graph. The intervals on the x axis are automatically calculated by dividing the total range of x values by the requested number of intervals (INTVLS).

The increment value of the y axis is determined by dividing the largest value to be graphed by the number of ROWS requested minus one. This creates a y axis with the specified number of rows ranging from 0 to the largest graphed value.

The significance line, indicated by the horizontal line of the character =, occurs at the average of all graphed values. This was requested by the keyword parameter LINE M.

The right side of the box defining the graph consists of the character I. This indicates that the graph is not continued on another page.

Plot

The plot routine provides a two-axis grid chart that graphs up to four fields as x coordinates against a fifth field (the y coordinates on the input file). Variables are plotted on the grid according to the coordinates received in the form of (y,x1,x2,x3,x4). Each x variable is represented by a different special character.

Syntax

```
%GRAPH1 infile PLOT parmfile parms control-field f1 f2 f3 f4 f5 f6 sortfield
%GRAPH2
```

`fn`

Plot uses the f1 through f5 parameters. Code F6 with zero. Do not use literals with the plot routine.

`sortfield`

The sort field must be the same as that specified for the y value.

Operation

The plot routine of the graphing facility uses the f1 parm to define the y value being plotted. The f2 through f5 parms are used to designate the fields to be plotted as x values. The y value should have an associated YDEF keyword and the x values should have associated XnDEF keywords in the Keyword File.

Keywords

The following keywords are used to format the plot graph. A coding example follows the keyword explanation.

1TITLE

2TITLE

3TITLE

FORMAT—
```
1TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
3TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

FUNCTION—Three optional nTITLE lines are allowed, each of which can contain up to 40 characters of titling information. Any or all lines can be omitted. The nTITLE line sequence numbers must be coded. Each nTITLE line results in one title line being printed on the graph.

DEFAULT—Output report title lines are blank if no titling information is entered.

ROWS

FORMAT—`ROWS nn`

FUNCTION—Specify the number of horizontal rows to be used per graph.

MINIMUM—15

MAXIMUM—40

DEFAULT—15

COLS

FORMAT—`COLS nn`

FUNCTION—Specify the number of columns to be used as the total width of the graph. If the number is less than the total combined width of the graph (based on the number and width of the bars requested), the system uses the output default value.

MINIMUM—32

MAXIMUM—99

DEFAULT—32

1FOOT

2FOOT

FORMAT—
```
1FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

FUNCTION—There are two optional nFOOT lines, each of which can contain up to 40 characters of alphanumeric information. These lines appear on the last page of output and are printed under the graph. The nFOOT sequence number must be coded.

DEFAULT—Blank lines

`X1DEF`

`X2DEF`

`X3DEF`

`X4DEF`

FORMAT—
```
X1DEF xxxxxxxxxxxx
X2DEF xxxxxxxxxxxx
X3DEF xxxxxxxxxxxx
X4DEF xxxxxxxxxxxx
```

FUNCTION—These keywords can contain up to 12 alphanumeric characters each. They cause a description of the specified x variables to be displayed on the printed output. One XnDEF keyword can be specified for each x variable reported.

DEFAULT—No x value descriptor items are printed. The XnDEF fields on the output are blank.

`YDEF`

FORMAT—`YDEF xxxxxxxxxxxx`

FUNCTION—This keyword can contain up to 12 alphanumeric characters. It causes a description of the specified y variable to be displayed on the printed output.

DEFAULT—No y value descriptor items are printed. The YDEF field on the output is blank.

`VARNUM`

FORMAT—`VARNUM n`

FUNCTION—Specify the number of x variables being used. The value of n must coincide with the number of XnDEF keywords used.

MINIMUM—1

MAXIMUM—4

DEFAULT—VARNUM is a required keyword. The plot routine does not execute if you omit VARNUM.

Example

The following is an example of a plot graph.

Input

```
FILE INFILE
REGION  1  4  N  0
AMTINV  5  8  P  2
AMTDUE  21  8  P  2
OVERDUE  29  4  N  0
FILE INCRD CARD
PRM 1 80 A
%GRAPH1 INFILE PLOT INCRD PRM REGION OVERDUE AMTINV AMTDUE 0 0 0 OVERDUE
%GRAPH2
END
1TITLE           SAMPLE FOR PLOT GRAPH
2TITLE             OVERDUE PAYMENTS
3TITLE      AMOUNT OF INVOICE AND AMOUNT DUE
ROWS 30
COLS 60
YDEF DAYS OVERDUE
X1DEF INVOICE AMT
X2DEF AMOUNT DUE
VARNUM 2
```

Output

```
                             SAMPLE FOR PLOT GRAPH
                                OVERDUE PAYMENTS
                        AMOUNT OF INVOICE AND AMOUNT DUE

                      @ VARIABLE DESCRIPTION: INVOICE AMT
                      & VARIABLE DESCRIPTION: AMOUNT DUE
                      Y VARIABLE DESCRIPTION: DAYS OVERDUE

     8.61 +---&-------+-&&--------@-----------+----@------+-------@---+
     8.31 !      &    !   &       !    @      !           !    @      !
     8.02 !           ! &&     @  !           ! @         !           !
     7.73 !  &       &!           !           !         @ !        @  !
     7.43 ! &        &! @         !           !           !         @ !
     7.14 +-----------+--&--------@-----------+-----------+-----------+
     6.85 !           &   @       !           !           !           !
     6.55 !           !   &       !      @    !           !           !
     6.26 !        &  !           !           !           ! @         !
     5.97 !           !           !           !           !           !
X    5.67 +-----------+-&-&--@----+----@------+-----------+-----------+
1    5.38 !           !           !           !           !           !
0    5.09 !    &     &!@          !           !         @ !      @    !
     4.79 !           !  &        @           !           !           !
+    4.50 &         &@!           !           !           !           @
0    4.21 +-----------+-----------+-----------+-----------+-----------+
1    3.91 !    &      !  &        !@          !           !           !
     3.62 !&          & &@    @   !           !           !          @!
     3.33 !         & !&  &       !   @     @ !     @     @           !
     3.03 !     &@ &@ &           !           !       @   !           !
     2.74 +-----------&-----------+-----------+--------@--+-----------+
     2.45 !           !           !           !           !           !
     2.15 !           &   @       !           !           !           !
     1.86 !           ! & &  @    !       @   !           !           !
     1.57 ! & &       !           !           !           !       @  @!
     1.27 +------&@&@-+-----------+-----------+-----------+-----------+
     0.98 !      &    !   &       !      @    !           !    @      !
     0.69 !  &        !&          !           !     @     !           !
     0.39 !      &@   !           !           !           !           !
     0.10 !           !   &       !   @       !           !           !
          +-----------+-----------+-----------+-----------+-----------+
         0.00        2.00        4.00        6.00        8.00       10.00
                          ( 3)
                       X10
```

The increment values of both the x and y axes are calculated by dividing the largest value to be graphed by the value specified for COLS minus one and ROWS minus one, respectively. This creates x and y axes with the specified number of columns and rows and with values ranging from 0 to the largest value graphed.

To keep the graph as simple as possible, quantitative values are factored. In the previous example, the y axis is labeled as X 10 +01, which indicates that the values displayed should be multiplied by 10, and the x axis is labeled as X 10 (03), which indicates that the values should be multiplied by 1000.

Deviation Bar Graph

The deviation bar graph supplies a graphic representation of the difference between the sums of two variables within control breaks. You can express this deviation as the numeric difference or as a percentage of difference.

Syntax

```
%GRAPH1 infile DEV parmfile parms control-field f1 f2 f3 f4 f5 f6 sortfield
%GRAPH2
```

`fn`

The deviation routine uses only the f1 and f2 parameters. You must code all others as zero. The f1 and f2 values are the fields being used as the x1 and x2 values in the deviation calculation. Do not use literals with the deviation routine.

`sortfield`

The sort field must be the same field named as the control field.

Operation

Values for each variable are accumulated until the value of the control field changes. At that time a control break occurs, and a horizontal bar is produced. This process is repeated for each value of the control field until end of file on the input file is reached.

On the deviation bar graph output, the control variable is listed as the left-hand scale. If, at a control break, the x2 variable is greater than the x1 variable, the horizontal bar representing their difference is on the negative side of the scale. If x2 is less than x1, the bar is graphed on the positive side.

The number of deviations that are graphed is determined by the number of control breaks encountered in the file. A maximum of 120 deviations is allowed. If more than 120 deviations are taken, an error message is generated, and processing stops.

Keywords

The following keywords are used to format the deviation bar graph. A coding example follows the keyword explanation.

`1TITLE`

`2TITLE`

`3TITLE`

FORMAT—
```
1TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
2TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
3TITLE xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

FUNCTION—Three optional nTITLE lines are allowed, each of which can contain up to 40 characters of titling information. Any or all lines can be omitted. The nTITLE line sequence numbers must be coded. Each nTITLE line results in one title line being printed on the graph.

DEFAULT—Output report title lines are blank if no titling information is provided.

`COLS`

FORMAT—`COLS nn`

FUNCTION—Specify the number of columns that will be reported on each half of the Deviation Bar Graph.

MINIMUM—1

MAXIMUM—40

DEFAULT—40

`LINE`

FORMAT—`LINE value`

FUNCTION—When you specify the LINE parameter, the graph prints an additional value, represented on the graph by a column of special characters (! and #) that bisects the graph at the location corresponding to its value within the deviation. This allows you to clearly see how your file deviates from a significant value. A valid value for the LINE parameter is an actual numeric value between –2,147,483,648 and 2,147,483,647.

For example, LINE 500000.00 results in a significant value of 500,000.00 being displayed as a vertical line on the graph. It is not necessary to code a decimal point when using whole numbers as significant values. (This example could have been coded LINE 500000.)

To produce a mean (or average) line of the values graphed, code an M after the LINE keyword. This automatically calculates and prints the mean value of the input file.

If the value of the significance line is out of range of the file being graphed, no significance value is reported.

DEFAULT—No line shown

1FOOT

2FOOT

FORMAT—`1FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`
`2FOOT xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx`

FUNCTION—You can specify two optional nFOOT lines, each of which can contain up to 40 characters of alphanumeric information. These lines appear on the last page of output and are printed under the graph. The nFOOT sequence number must be coded.

DEFAULT—Blank lines

YVARS

FORMAT—`YVARS xxxxxxxx`

FUNCTION—This optional field describes the control field being used. It can contain up to eight alphanumeric characters.

DEFAULT—No control value description is printed. When this keyword is omitted, the control value field on the printed graphic output is blank.

PCTDEV

FORMAT—`PCTDEV n`

FUNCTION—Use this parameter to specify which type of deviation will be graphed. There are two options for n. A zero indicates that the difference will be graphed, and a one indicates that the difference as a percentage is graphed.

DEFAULT—0

Example

This is an example of the deviation bar graph.

Input

```
FILE INFILE
SALESMAN  1  3  A
SALESYTD  4  8  P  2
AMTDUE   12  8  P  2
REGION   20  3  N
FILE INCRD CARD
PRM 1 80 A
%GRAPH1 INFILE DEV INCRD PRM SALESMAN SALESYTD AMTDUE 0 0 0 0 SALESMAN
%GRAPH2
END
1TITLE    SAMPLE FOR DEVIATION BAR GRAPH
2TITLE       SALES-YEAR-TO-DATE MINUS
3TITLE     AMOUNT DUE FOR EACH SALESMAN
LINE M
COLS 30
YVARS SALESMAN
PCTDEV 0
```

Output

```
                          SAMPLE FOR DEVIATION BAR GRAPH
                          SALES-YEAR-TO-DATE MINUS
                          AMOUNT DUE FOR EACH SALESMAN


                ALL VALUES HAVE BEEN SCALED BY A FACTOR OF        100


                EACH X REPRESENTS    95.80

         +--------------------------------------------------------------+
         I -2874.10  -1916.07   -958.03 -+   958.03   1916.07   2874.10 I
         I *         *         *                  *         *         * I
AD       I                              XXXXXXXXXXXX#XXXXXXXX           I
CD       I                              XXXXXXXXXXXX!                   I
ED       I                          XXXX            !                   I
FS       I                              XXXXXXXXXXXX#XX                 I
GE       I              XXXXXXXXXXXXXXXXX           !                   I
JD       I                              XXXXXXXXXXXX#XXXXXXXXXX         I
SA       I                            XX            !                   I
SF       I                              XXXXXXXXXXXX#XXX                I
TD       I                              XXXXXXXXXXXX#XXXXXXXXXXXXXXXXX  I
WS       I                              XXXXXXXXXXXX#XXXXXXXXXXXXX      I
         I *         *         *                  *         *         * I
         I -2874.10  -1916.07   -958.03 -+   958.03   1916.07   2874.10 I
         +--------------------------------------------------------------+

CONTROL VARIABLE = SALESMAN
```

This example demonstrates the difference option of the deviation bar graph, which is specified by the PCTDEV 0 keyword. For each salesman listed on the y axis, the value for amount due is subtracted from sales-year-to-date and is represented by a horizontal bar graph.

The significance line, indicated by the vertical line of the characters ! and #, occurs at the average of all graphed values. This was requested by the keyword parameter LINE M. The character ! is printed as the significance line when the position is blank. When the position contains the character X, the character # is printed for the significance line.

To keep the graph as simple as possible, quantitative values are factored so that values printed along the x axis do not become too large. In the example, the values printed are factored by 100. This function is performed automatically, and the factor varies depending on the size of the input values.

Example OS/390 and z/OS JCL

The OS/390 and z/OS JCL required to execute the graphing routines is the same as required to execute any CA-PanAudit Plus program, with the addition of the following DD card:

```
//OUTPUT DD SYSOUT=A (data set used for output)
```

The exact format of this card may vary slightly. The following example shows JCL, the required invocation statements, and appropriate keywords.

```
//jobname  JOB  accounting.info
//STEPNAME EXEC PGM=EZTPA00,REGION=512K
//STEPLIB  DD   DSN=your.caeztpls.loadlib,DISP=SHR
//         DD   DSN=your.capaupls.loadlib,DISP=SHR

//PANDD    DD   DSN=your.capaupls.macro.library,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSSNAP  DD   SYSOUT=*
//SYSOUT   DD   SYSOUT=*
//EZTVFM   DD   UNIT=SYSDA,SPACE=(CYL,(10,2))
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(15,5))
//SORTWK02 DD   UNIT=SYSDA,SPACE=(CYL,(15,5))
//SORTWK03 DD   UNIT=SYSDA,SPACE=(CYL,(15,5))
//INREC    DD   ...
//SYSIN    DD   *
FILE INREC
   SALESYTD  1  8  N  2
   COMMYTD   9  8  N  2
   REGION   49  4  A
FILE INCRD CARD
PRM  1  80  A
%GRAPH1 INREC BAR INCRD PRM REGION SALESYTD COMMYTD 0 0 0 0 REGION
%GRAPH2
END
1TITLE    SAMPLE TITLE LINES FOR
2TITLE TEST OF THE STANDARD BAR GRAPH
LINE 150000
FLD1,VAR,MAX
FLD2,VAR,MEAN
BARKEY1 SALESYTD MAX
BARKEY2 COMMYTD MEAN
ROWS 40
COLS 99
WIDTH 10
SPACE 3
BARS 2
1FOOT TEST FOOTER FOR SAME REASON
/*
//
```

Chapter

12 Advanced Techniques

Due to the macro-based design of CA-PanAudit Plus, you can design sophisticated auditing applications without learning complicated programming languages, syntax, or procedures.

However, because of the advanced power of the host language, CA-Easytrieve Plus , and the flexibility of the design of CA-PanAudit Plus routines, you can create complex applications. This chapter describes some of the advanced techniques that you can use with CA-PanAudit Plus routines. These advanced techniques are based on the features of CA-Easytrieve Plus .

This chapter covers the following advanced techniques:

- Use of CA-Easytrieve Plus procedures
- Stand-alone DISPLAY and stand-alone REPORT routines
- Logic-after-invocation section of stand-alone routines
- Logic-before-invocation section of stand-alone routines
- Use of sample flags
- Stacking stand-alone routines
- Database use of CA-PanAudit Plus
- Miscellaneous techniques

The end of this chapter contains a functional chart of inline and stand-alone routines and a summary of the advanced techniques.

CA-Easytrieve Plus Procedures

CA-PanAudit Plus routines are written in CA-Easytrieve Plus , an information retrieval and data management system. One of the features of CA-Easytrieve Plus is the use of procedures for defining your own routines within a program. By defining your own routines within a program, logic flow is simplified.

Define procedures within a program with the PROC and END-PROC keywords and invoke procedures with the PERFORM statement (see the CA-Easytrieve Plus *Reference Guide* for details). For example, consider the following code which demonstrates the use of procedures to simplify logic flow:

```
   ...
JOB INPUT ...
PERFORM TASK1
PERFORM TASK2
PERFORM TASK1
PERFORM TASK3
PERFORM TASK1
PERFORM TASK4
*
TASK1. PROC
   .
   .
   .
END-PROC
*
TASK2. PROC
   .
   .
   .
END-PROC
*
TASK3. PROC
   .
   .
   .
END-PROC
*
TASK4. PROC
   .
   .
   .
END-PROC
```

In this example, the PROC and END-PROC statements are used to define four procedures. The PERFORM statement invokes these procedures. Performing these procedures eliminates the need for repeating sections of code, branching statements, or other complex logic that inline coding techniques require. Instead, the procedures define routines within the main routine. The PERFORM statement can invoke these procedures any time.

Stand-alone CA-PanAudit Plus routines extensively use this method of coding, called structured coding. You should use structured coding techniques although the same results can be obtained with inline techniques. This is because programs are easier to write and debug when they use structured techniques.

Procedures are an integral part of structured coding, and you can use them in many areas of CA-PanAudit Plus, such as:

- With the logic-before-invocation section of some stand-alone routines
- With the logic-after-invocation section of some stand-alone routines
- With inline routines
- With REPORT subactivities
- With the sampling flags technique

The next example demonstrates the use of procedures in a CA-PanAudit Plus routine.

Example Procedure with an Inline Routine

```
FILE PAYFILE
   EMPLOYEE-NAME       1   20   A
   EMPLOYEE-CODE      21    1   A
   MEDICAL-DEDUCT     23    3   P  2
   SSN                31    9   N
*
JOB INPUT PAYFILE
PERFORM SCREEN-FILE
PERFORM VALIDATE
*
SCREEN-FILE. PROC
IF EMPLOYEE-CODE EQ 'P'
   GO TO JOB
END-IF
END-PROC
*
VALIDATE. PROC
%NUMTEST MEDICAL-DEDUCT 'MEDICAL-DEDUCT NOT NUMERIC' +
         EMPLOYEE-NAME
IF NUMTEST-FLAG EQ 'YES' AND MEDICAL-DEDUCT NE 0
   PRINT VALID-MED
END-IF
END-PROC
*
REPORT VALID-MED
TITLE 01 'EMPLOYEES WITH MEDICAL DEDUCTIONS'
LINE EMPLOYEE-NAME SSN MEDICAL-DEDUCT
```

The code in this sample demonstrates how procedures are used with the inline routine, NUMTEST. The first portion of code is the library section. The next section of code is the JOB INPUT PAYFILE statement. It identifies the automatic input file and also contains two PERFORM statements which execute the procedures.

The SCREEN-FILE procedure executes the GO TO JOB statement if the employee is part time. The VALIDATE procedure checks for a valid numeric value in the MEDICAL-DEDUCT field. If the value is valid and nonzero, the VALIDATE procedure prints a line of the VALID-MED report.

Note the order of the various sections of code:

1. Library section
2. Mainline code – JOB INPUT and PERFORM statements
3. Definition of procedures
4. Definition of REPORT

For complete details regarding the correct order of statements, see the CA-Easytrieve Plus *Reference Guide.*

Procedure Placement

Place CA-Easytrieve Plus procedures immediately after the associated activity (JOB or SORT) or subactivity (REPORT).

Place procedures that you use with stand-alone routines after the CA-PanAudit Plus macro in the logic-after-invocation section or before the CA-PanAudit Plus macro in the logic-before-invocation section. Do not place procedures between the first and second macro invocations. For more details, see Stand-alone Routines (DISPLAY and REPORT).

Stand-alone Routines (DISPLAY and REPORT)

Stand-alone DISPLAY and stand-alone REPORT routines are two distinct types of CA-PanAudit Plus routines. However, in most applications, the difference between these two types of stand-alone routines is transparent. The only time that their difference is important is when an application uses the logic-after-invocation section of stand-alone routines. These two subjects, Stand-alone DISPLAY and Stand-alone REPORT routines and logic-after-invocation in Stand-alone routines, are related and are discussed together in this section.

Logic-after-invocation of Stand-alone Routines

Stand-alone display and stand-alone report routines are differentiated by the method that they use to create a listing. The stand-alone display routine uses the CA-Easytrieve Plus DISPLAY statement, and the stand-alone report routine uses a REPORT subactivity to create listings. Because of the different requirements of DISPLAY and REPORT statements, there is a corresponding difference in the use of the logic-after-invocation section for these routines.

The four types of CA-Easytrieve Plus statements that you can code in the logic-after-invocation section are:

- Procedures
- JOB or SORT activities
- REPORT statements
- Special-name REPORT procedures

Stand-alone REPORT

Stand-alone REPORT routines cannot define a procedure in the logic-after-invocation section. This is because of the REPORT statement at the end of all stand-alone REPORT routines. A procedure must be placed immediately after the associated activity. Therefore, a procedure in the logic-after-invocation section of a stand-alone REPORT routine is associated with the REPORT subactivity, not the previous JOB or SORT activity. The only procedure that can follow a stand-alone REPORT routine is a special-name REPORT procedure.

Stand-alone REPORT routines cannot define a REPORT statement in the logic-after-invocation section. This is because of the structure used in stand-alone REPORT routines. Since most stand-alone REPORT routines use multiple CA-Easytrieve Plus activities, the screening code section, in which the report is printed, is associated with a different activity from where the REPORT statement is placed.

Stand-alone DISPLAY

A stand-alone DISPLAY routine can use a procedure in the logic-after-invocation section, because a stand-alone DISPLAY routine does not use a REPORT subactivity. The remaining two types of CA-Easytrieve Plus statements, which can be associated with both stand-alone DISPLAY and stand-alone REPORT routines, are JOB or SORT activities and special-name REPORT procedures.

JOB or SORT Activities

A JOB or SORT activity defines a new activity that is not directly related to the CA-PanAudit Plus routine. This new activity can take the form of a user-coded activity or an activity encountered through the coding of an additional CA-PanAudit Plus routine (see Stacking Stand-alone Routines later in this chapter).

Special-name REPORT Procedures

You can also use the logic-after-invocation section with special-name REPORT procedures.

The REPORT subactivity has special-name REPORT procedures that you can define to perform various functions. When you code a special-name procedure, the logic within the procedure is performed at the appropriate time for its function (see the CA-Easytrieve Plus *Reference Guide*).

For example, when you code the ENDPAGE procedure, it executes every time CA-PanAudit Plus reaches the end of report body. You can use this procedure to produce footers at the bottom of each page of a report. For a list and description of these procedures, see the CA-Easytrieve Plus *Reference Guide.*

You can code special-name REPORT procedures in both types of stand-alone routines. Since stand-alone report routines always end with a REPORT, code the special-name REPORT procedure immediately after the second macro invocation. Since stand-alone display routines do not contain a REPORT, use stand-alone display routines only if you define a REPORT in the logic-after-invocation section.

Some stand-alone REPORT routines use special-name REPORT procedures. When this occurs, you cannot define that special-name REPORT procedure in the logic-after-invocation section of the stand-alone REPORT routine. This is because special-name REPORT procedures can only be used once for a given report. The following lists these stand-alone REPORT routines and the special-name REPORT procedures that they use:

```
Routine     Special-name REPORT Procedures Used

AGING       AFTER-BREAK, TERMINATION
INTERVL     BEFORE-BREAK, TERMINATION
OCCURS      BEFORE-BREAK
STRATIF     BEFORE-BREAK, AFTERBREAK
VERSUS      BEFORE-BREAK, TERMINATION
```

Note: If a macro contains more than one report, any report procedure added by the client will only affect the last report defined in the macro. Review macro processing to determine if and when the last report will be printed.

Additional examples of the use of procedures in CA-PanAudit Plus are shown in subsequent topics in this chapter.

Placement of Procedures or Reports

The method that you use to code procedures and REPORTs in the logic-after-invocation section are identical. You perform a procedure or print a REPORT in the screening code section. Then, code the procedure that you want to perform, or the REPORT that you want to print, in the logic-after-invocation section of the stand-alone routine. If both procedures and REPORTs are used with the stand-alone DISPLAY routine, the statements defining the procedures must precede the REPORT statements. This is because the REPORT and any associated REPORT procedures must be the last statements in any CA-Easytrieve Plus job activity.

You must also code special-name REPORT procedures in the logic-after-invocation section. These special procedures must immediately follow the associated REPORT. If a program contains a procedure, REPORT, and a special-name REPORT procedure, the order of occurrence must be:

1. Procedure
2. REPORT
3. Special-name REPORT procedure

The special-name REPORT procedure is the only type of procedure that you can specify in the logic-after-invocation section of a stand-alone REPORT routine. Example Three - Special-name REPORT procedure demonstrates this combination.

Example One — Procedure in a Stand-alone-DISPLAY Routine

```
FILE PAYFILE
   EMPLOYEE-CODE     5    1    A
   GROSS-PAY        23    5    P  2
   HIRE-DATE        31    5    N
FILE SAMPFIL
*
%DOLUNIT1 PAYFILE GROSS-PAY 10000 2000 13453
PERFORM SCREEN1
PERFORM SCREEN2
%DOLUNIT2 SAMPFIL
*
SCREEN1. PROC
IF EMPLOYEE-CODE EQ 'P'
   GO TO JOB
END-IF
END-PROC
*
SCREEN2. PROC
%DAYSAGO HIRE-DATE YYDDD LT 30
IF DAYSAGO-FLAG EQ 'YES'
   GO TO JOB
END-IF
END-PROC
```

Instead of coding logic in the screening code section, the example PERFORMs a procedure. The logic-after-invocation section defines the procedure, and the procedure performs the same screening function as the example in the chapter "Coding Guidelines." The logic-after-invocation section can define a procedure because DOLUNIT is a stand-alone DISPLAY routine and does not use a REPORT subactivity.

The PERFORM SCREEN1 statement invokes the SCREEN1 procedure. The SCREEN1 procedure bypasses the processing of records having an EMPLOYEE-CODE of P. The PERFORM SCREEN2 statement invokes the SCREEN2 procedure. The SCREEN2 procedure invokes the inline routine DAYSAGO to determine if the HIRE-DATE is less than 30 days old. If this is true, then the GO TO JOB statement bypasses the record.

Example Two — Procedure and REPORT in a Stand-alone-DISPLAY Routine

```
FILE PAYFILE
   EMPLOYEE-NAME     1  20   A
   GROSS-PAY        23   5   P  2
   SSN              31   9   N
FILE SAMPFIL
*
%DOLUNIT1 PAYFILE GROSS-PAY 10000 2000 13453
PERFORM CUTOFF-CHECK
%DOLUNIT2 SAMPFIL
*
CUTOFF-CHECK. PROC
IF GROSS-PAY GE 2000
   PRINT TOP-ITEMS
END-IF
END-PROC
*
REPORT TOP-ITEMS
SEQUENCE EMPLOYEE-NAME
TITLE 01 'EMPLOYEES WITH GROSS PAY EXCEEDING CUTOFF'
LINE EMPLOYEE-NAME SSN GROSS-PAY
```

Since DOLUNIT is a stand-alone DISPLAY routine, procedures can be coded in the logic-after-invocation section. In this example, the CUTOFF-CHECK procedure is PERFORMed in the screening code section. The CUTOFF-CHECK procedure checks to see if GROSS-PAY is greater than or equal to the cutoff value, 2000. If this condition is true, a line of the TOP-ITEMS REPORT is printed. Since both a procedure and a REPORT are used, the statements defining the procedure occur before the REPORT statements.

Example Three — Special-name REPORT Procedure

The following example demonstrates the use of a special-name REPORT procedure with a stand-alone REPORT routine:

```
FILE CUSTFIL...
  CUSTNO      1    6   N
  INVNO      11    6   N
  BALANCE    17    6   P  2
  DATE       23    6   N
  ...
%AGING1 CUSTFIL DATE MMDDYY BALANCE
%AGING2 CUSTNO INVNO CURANGE
*
ENDPAGE. PROC
  DISPLAY +20 'CONFIDENTIAL  FOR AUDITORS ONLY'
END-PROC
```

In this example the ENDPAGE REPORT procedure is used to display a footer at the bottom of the report. The REPORT subactivity automatically performs the ENDPAGE procedure when the bottom of the report body is reached. It is not necessary for you to perform any of the special-name REPORT procedures. They are performed at the appropriate time for their function as the CA-Easytrieve Plus *Reference Guide* describes.

Logic-before-invocation of Stand-alone Routines

The logic-before-invocation section in stand-alone routines allows you to define CA-Easytrieve Plus activities prior to the invocation of the CA-PanAudit Plus routine. This allows you to define multiple activities in a single execution of CA-PanAudit Plus.

Statements that you code in the logic-before-invocation section must contain a complete CA-Easytrieve Plus activity (JOB or SORT) and may contain a REPORT subactivity. Common examples of activities that you can perform in the logic-before-invocation section are:

- Pre-calculation techniques
- Creating an input file from two or more files

The logic-before-invocation section is located after the library section and prior to the first macro invocation. This means that any input or output file for the logic-before-invocation section must be defined in the library section. (There can be only one library section in a single execution of CA-PanAudit Plus.) If you want to screen, reconstruct, or merge an input file or files in the logic-before-invocation section, you must create a new temporary or permanent file. Specify the temporary or permanent file as input to the CA-PanAudit Plus routine. The following examples demonstrate this technique.

Limitation

Some stand-alone routines contain FILE statements that do not allow you to use the logic-before-invocation section. The beginning of the CA-PanAudit Plus routine (which follows the library section) contains these FILE statements. With these routines it is not possible to code a CA-Easytrieve Plus activity between the library section and macro invocation statement because the code would be placed in the middle of the library section.

The following is a list of routines that do not support the logic-before-invocation section:

```
ADDRCMP      MULTDUP
CAVEVAL      SRCECOMP
CBLCNVRT     STOPORGO
DUPTEST      STRATIF
ENCRYPT      STRTEVL
INTERVL      VERSUS
```

The following examples demonstrate techniques for using the logic-before-invocation section.

Example One — Pre-calculation Technique

```
FILE INFILE ...
FILE SAMPFIL ...
*
%POPSIZE1 INFILE
%POPSIZE2
*
%RANDPCT1 INFILE POPSIZE 2.0 984875
%RANDPCT2 SAMPFIL
```

The RANDPCT routine requires an exact count of the population size as a parameter. One method of obtaining this information is to run an INTERVL analysis, or another CA-Easytrieve Plus program, to obtain the record count. Then, put the value of the record count into the parameter list. This method is time-consuming and requires an extra execution of CA-PanAudit Plus to obtain the population size.

Example One demonstrates a pre-calculation technique that automatically supplies the population size to RANDPCT. The logic-before-invocation section consists of the CA-PanAudit Plus routine POPSIZE. POPSIZE finds the population of the file and assigns it to the field POPSIZE. POPSIZE is then specified as the size parameter in the RANDPCT routine.

The pre-calculation technique ensures that the exact population size is always specified in the appropriate sampling routines. Use this technique for all sampling routines that require an exact population size as a parameter.

Example Two — Creating an Input File from Two or More Files

```
FILE REGSMP
   CUSTOMER-ID            1   6   N
   AMOUNT-DUE            27   6   P  2
FILE REGAUD
   AUD-CUST-ID            1   6   N
   AUD-AMT-DUE            7  10   N  2
FILE AUDSMP
   AUDSMP-CUST-ID         1   6   N
   REC-AMT                7   6   P  2
   AUD-AMT               13   6   P  2
DEFINE REPLIES-USED       S   5   P  0    VALUE (0)
DEFINE REPLIES-NOT-USED   S   5   P  0    VALUE (0)
DEFINE STOP-FLAG          S   3   A       VALUE ('NO')
*
JOB INPUT (REGSMP KEY CUSTOMERID    +
           REGAUD KEY AUDCUSTID)  FINISH END-OF-JOB
IF NOT MATCHED AND REGAUD
   STOP-FLAG = 'YES'
   PRINT NOMATCH
   GO TO JOB
END-IF
IF MATCHED
   AUD-AMT = AUD-AMT-DUE
   REPLIES-USED = REPLIES-USED + 1
END-IF
IF NOT MATCHED AND REGSMP
   AUD-AMT = AMOUNT-DUE
   REPLIES-NOT-USED = REPLIES-NOT-USED + 1
END-IF
REC-AMT = AMOUNT-DUE
AUDSMP-CUST-ID = CUSTOMER-ID
PUT AUDSMP
END-OF-JOB. PROC
PRINT REPLIES
IF STOP-FLAG EQ 'YES'
   STOP-EXECUTE
END-IF
END-PROC
REPORT NO-MATCH
TITLE 01 'NON-MATCHING CUSTOMER IDS'
LINE 01 AUD-CUST-ID
REPORT REPLIES
TITLE 01 'SUMMARY OF CUSTOMER REPLIES IN AUDSMP FILE'
LINE 01 REPLIES-USED REPLIES-NOT-USED
%REGSAM1 AUDSMP REC-AMT AUD-AMT 32555
%REGSAM2
```

Example Two demonstrates the creation of the input file to REGSAM from the data of two separate files. The REGSMP file is the preliminary sample file to REGSAM. The REGAUD file is created using information from customer replies to confirmation letters regarding the accuracy of the data in the preliminary sample file. After the execution of the code, the AUDSMP file contains the proper recorded and audited amounts obtained from the REGSMP and REGAUD files.

This job performs synchronized file processing. The JOB INPUT statement defines the two files to be processed and the keys that are to be matched (see the CA-Easytrieve Plus *Reference Guide*).

The first IF statement checks for a nonmatched condition with the extra record being in the REGAUD file. If this is true, a record exists on the audited file with no match on the sample file. This is an error condition and means that an incorrect customer ID was entered when the REGAUD file was created. In this case, the job sets the STOP-FLAG to 'YES' and prints the nonmatching customer ID. The GO TO JOB statement bypasses the processing of this record.

The next IF statement checks for a matched condition. If this is true, an audited AMOUNT-DUE was obtained from a customer. The job assigns this amount to the AUD-AMT field in the AUDSMP file. Also, the job increments a counter named REPLIES-USED to keep track of the number of customer replies in the AUDSMP file.

The final IF statement checks for a nonmatched condition with the extra record being in the REGSMP file. If this is true, a record exists on the sample file with no match on the audited file. This condition is acceptable and means that the customer did not reply to the confirmation letter with an audited amount. In this case, it is assumed that the audited amount is the same as the recorded amount, and the job makes the appropriate assignment. Also, the job increments a counter named REPLIES-NOT-USED to keep track of the number of nonreplies in the AUDSMP file. Then, the job assigns the AMOUNT-DUE field to the REC-AMT field for processing by the subsequent REGSAM routine. The job then writes the record to the AUDSMP file.

When both files reach the end of file, the END-OF-JOB procedure is automatically performed. This procedure prints the summary of customer replies and checks the STOP-FLAG for a value of YES. If this condition is true, the previously described error condition was encountered, and the STOP-EXECUTE statement prevents the execution of REGSAM. Otherwise, the REGSAM routine is executed.

Use of Sample Flags

CA-PanAudit Plus routines that create sample files use sample flags. Sample flags are reserved fields indicating whether a given record was selected for the sample file. If the reserved field contains the value YES, the record was selected for the sample file; if it contains NO, the record was not selected.

The PERFORM procname parameter identifies the routines that use the sample flag technique. The procname is a procedure that the routine PERFORMs after the value in the reserved field is set to YES or NO. This provides the opportunity to define a procedure in the logic-after-invocation section to print a report or perform other activities based on the selection of records for the sample population.

The following routines use the sample flag technique:

```
ATTSAMP          RANDPCT
CAVSAMP          RANDXCT
DISCSMP          REGSAMP
DOLUNIT          SPS
EACHNTH          STOPORGO
INTSAMP          VARSAMP
```

The sample flag technique allows you to perform various functions for both positive and negative sampling. You can print listings of records selected or records not selected. It is also possible to create a file of records not selected or perform complex processing logic such as creating reformatted files.

Since you apply the sample flag technique by coding a procedure, there is great flexibility in the type of results that you can obtain. The following examples illustrate three different methods of using sample flags to create listings.

Example One — Sample Flags with DISPLAY

Input

```
FILE PAYFILE ...
   NET              5    6   P  2
   EMPLOYEE-NAME   24   20   A
   ...
FILE SAMPFIL ...
*
%SPS1 PAYFILE NET 2000 93848
%SPS2 SAMPFIL PERFORM SELECT-LIST
SELECT-LIST. PROC
   IF SPS-SELECTED EQ 'YES'
      DISPLAY 'SELECTED' +2 EMPLOYEE-NAME
   END-IF
END-PROC
```

This example demonstrates a sampling proportional to size algorithm on a net pay field in a payroll file. The PERFORM SELECT-LIST parameter invokes the sample flag technique. This tells the SPS routine to set the reserved field, SPS-SELECTED, to the value YES or NO, depending on whether a given record is selected for the sample file. The routine then performs the SELECT-LIST procedure.

The user codes the SELECT-LIST procedure in the logic-after-invocation section of the program. This procedure tests the SPS-SELECTED field for the value YES. If this condition is true, then the word SELECTED and the employee's name are printed.

Output

```
                    SPS SAMPLING REPORT
                      INPUT PARAMETERS

        INPUT FILENAME                              INFILE
        INPUT FIELD                                    NET
        VALUE OF INPUT FIELD IS                        ABS
        TARGET VALUE                             20,000.00

                      SAMPLE FILE
SELECTED  EMPLOYEE37
SELECTED  EMPLOYEE76
SELECTED  EMPLOYEE112
SELECTED  EMPLOYEE146
SELECTED  EMPLOYEE186
SELECTED  EMPLOYEE224
SELECTED  EMPLOYEE263
SELECTED  EMPLOYEE301
SELECTED  EMPLOYEE336
SELECTED  EMPLOYEE376
SELECTED  EMPLOYEE420
SELECTED  EMPLOYEE456
SELECTED  EMPLOYEE494
SELECTED  EMPLOYEE535
SELECTED  EMPLOYEE572
SELECTED  EMPLOYEE606
SELECTED  EMPLOYEE641
SELECTED  EMPLOYEE682
SELECTED  EMPLOYEE718
SELECTED  EMPLOYEE755
SELECTED  EMPLOYEE793
SELECTED  EMPLOYEE833
SELECTED  EMPLOYEE872
SELECTED  EMPLOYEE912

        NUMBER OF RECORDS PROCESSED                    928
        ABS VALUE OF RECORDS PROCESSED          488,886.12
        ACT VALUE OF RECORDS PROCESSED          488,886.12
        POS VALUE OF RECORDS PROCESSED          488,886.12
        NUMBER OF RECORDS IN SAMPLE FILE                24

        FILE SAMPFIL WILL BE CREATED
```

The output list shows the input parameters and the beginning of the sample file statistics. However, the list of selected employees is printed before the sample file results are printed. This is because the SELECT-LIST procedure contains a DISPLAY statement that immediately prints a line. These lines of output occur before the sample file statistics because the statistics are printed after the sampling of all records.

Example Two — Sample Flags with REPORT

Input

```
FILE PAYFILE ...
   NET              5    6   P  2
   EMPLOYEE-NAME    24   20   A
   ...
FILE SAMPFIL ...
*
%SPS1 PAYFILE NET 2000 93848
%SPS2 SAMPFIL PERFORM SELECT-LIST
SELECT-LIST. PROC
   IF SPS-SELECTED EQ 'YES'
      PRINT SELECT-RPT
   END-IF
END-PROC
*
REPORT SELECT-RPT
TITLE 01 'SPS SELECTION LIST OF PAYFILE'
LINE EMPLOYEE-NAME NET
```

This example performs the same function as Example One, but instead of using a DISPLAY statement to produce the selection list, a REPORT statement is used. The PRINT SELECT-RPT statement replaces the DISPLAY statement, and a line of the report is generated each time the SPS-SELECTED field contains the value YES.

The report specifies a title and lists each employee's name and their associated net pay. The logic-after-invocation section contains both a procedure and a REPORT, with the procedure preceding the REPORT statements.

Output

```
              SPS SAMPLING REPORT
                INPUT PARAMETERS

  INPUT FILENAME                              INFILE
  INPUT FIELD                                    NET
  VALUE OF INPUT FIELD IS                        ABS
  TARGET VALUE                                20,000.00

                SAMPLE FILE
                           SPS SELECTION LIST OF PAYFILE                        PAGE      1

                                         EMPLOYEE-NAME            NET

                                         EMPLOYEE37                    165.34
                                         EMPLOYEE76                    374.71
                                         EMPLOYEE112                   952.28
                                         EMPLOYEE146                   396.86
                                         EMPLOYEE186                   228.74
                                         EMPLOYEE224                   504.76
                                         EMPLOYEE263                   748.33
                                         EMPLOYEE301                   779.90
                                         EMPLOYEE336                   953.15
                                         EMPLOYEE376                   797.40
                                         EMPLOYEE420                   701.52
                                         EMPLOYEE456                   869.73
                                         EMPLOYEE494                   483.01
                                         EMPLOYEE535                   752.38
                                         EMPLOYEE572                   171.24
                                         EMPLOYEE606                   962.78
                                         EMPLOYEE641                   626.59
                                         EMPLOYEE682                   937.00
                                         EMPLOYEE718                   480.92
                                         EMPLOYEE755                   388.09
                                         EMPLOYEE793                   428.20
                                         EMPLOYEE833                   295.36
                                         EMPLOYEE872                   844.52
                                         EMPLOYEE912                   821.08

   NUMBER OF RECORDS PROCESSED                    928
   ABS VALUE OF RECORDS PROCESSED             488,886.12
   ACT VALUE OF RECORDS PROCESSED             488,886.12
   POS VALUE OF RECORDS PROCESSED             488,886.12
   NUMBER OF RECORDS IN SAMPLE FILE                24

   FILE SAMPFIL WILL BE CREATED
```

This example shows that the results of this method are similar to those shown in the previous one. The input parameters are listed, and the sample file results are interrupted by the selection list report.

If you do not want the selection list to occur in the middle of the SPS report, the next example shows how you can separate these listings.

Example Three — Sample Flags with REPORT

Input

```
FILE PAYFILE ...
   NET              5    6   P  2
   EMPLOYEE-NAME   24   20   A
   ...
FILE SAMPFIL ...
*
%SPS1 PAYFILE NET 2000 93848
%SPS2 SAMPFIL PERFORM SELECT-LIST
SELECT-LIST. PROC
   IF SPS-SELECTED EQ 'YES'
      PRINT SELECT-RPT
   END-IF
END-PROC
*
REPORT SELECT-RPT
SEQUENCE EMPLOYEE-NAME
TITLE 01 'SPS SELECTION LIST OF PAYFILE'
LINE EMPLOYEE-NAME NET
```

This example is identical to the one shown in Example 2 with one exception. The report contains a SEQUENCE EMPLOYEE-NAME statement. This sequencing forces the output data to be spooled to temporary storage for sequencing. The storage required to spool and sequence the records is obtained automatically.

Due to the sequencing, a line of the report does not immediately print each time SPS-SELECTED contains the value YES. Instead, CA-Easytrieve Plus spools a line of output to temporary storage, and it remains there until all records have been processed. Then CA-Easytrieve Plus sequences the output records by EMPLOYEE-NAME as specified. This allows the SPS routine to finish processing and complete printing before the SELECT-RPT is printed.

Output

```
               SPS SAMPLING REPORT
                INPUT PARAMETERS

INPUT FILENAME                              INFILE
INPUT FIELD                                    NET
VALUE OF INPUT FIELD IS                        ABS
TARGET VALUE                              20,000.00

                 SAMPLE FILE

NUMBER OF RECORDS PROCESSED                    928
ABS VALUE OF RECORDS PROCESSED          488,886.12
ACT VALUE OF RECORDS PROCESSED          488,886.12
POS VALUE OF RECORDS PROCESSED          488,886.12
NUMBER OF RECORDS IN SAMPLE FILE                24

FILE SAMPFIL WILL BE CREATED
```

```
SPS SELECTION LIST OF PAYFILE                  PAGE      1

      EMPLOYEE-NAME            NET

     EMPLOYEE112                   952.28
     EMPLOYEE146                   396.86
     EMPLOYEE186                   228.74
     EMPLOYEE224                   504.76
     EMPLOYEE263                   748.33
     EMPLOYEE301                   779.90
     EMPLOYEE336                   953.15
     EMPLOYEE37                    165.34
     EMPLOYEE376                   797.40
     EMPLOYEE420                   701.52
     EMPLOYEE456                   869.73
     EMPLOYEE494                   483.01
     EMPLOYEE535                   752.38
     EMPLOYEE572                   171.24
     EMPLOYEE606                   962.78
     EMPLOYEE641                   626.59
     EMPLOYEE682                   937.00
     EMPLOYEE718                   480.92
     EMPLOYEE755                   388.09
     EMPLOYEE76                    374.71
     EMPLOYEE793                   428.20
     EMPLOYEE833                   295.36
     EMPLOYEE872                   844.52
     EMPLOYEE912                   821.08
```

The output shows the full SPS listing, followed by the employee selection list. The sequencing of the report causes the separation of the listings.

Stacking Stand-alone Routines

In some applications, you can code multiple stand-alone routines in a single CA-PanAudit Plus execution. This is called stacking of routines. The stacking of routines provides a technique to decrease execution time, reduce the coding of JCL, eliminate the repeated submission of jobs, plus define and create customized complex applications. When you stack stand-alone routines, successive JOB and/or SORT activities are combined after a single library section. This is a form of logic-after-invocation in a stand-alone routine. However, there is a difference between how you can stack and combine inline and stand-alone routines.

The chapter "Coding Guidelines" explains how you can use more than one inline routine in a single execution and how you can use inline routines with stand-alone routines. These are basic topics that you can uniformly apply to inline and stand-alone routines and are fully discussed in this chapter. However, the stacking of stand-alone routines is a complex issue and is discussed in detail later. Certain limitations exist in stacking stand-alone routines, and these limitations must be fully understood before you use this technique.

Inline routines do not use input files, do not contain CA-Easytrieve Plus REPORT subactivities, and do not produce output files. Because inline routines do not have these processing capabilities, you are able to stack inline routines and combine them with stand-alone Routines O-R any valid CA-Easytrieve Plus logic.

On the other hand, all stand-alone routines use input files, and some use REPORT subactivities and produce output files. Some even define their own files for internal use and must, therefore, contain a library section. These processing capabilities make it more difficult to provide the unrestricted flexibility that is available with inline routines.

Limitations

Use the following guidelines when stacking stand-alone routines:

- Certain stand-alone routines can only be the first of any stacked stand-alone routines. They cannot exist as second or subsequent stand-alone routines.
- When you repeat some stand-alone routines, you cannot change certain parameters in the second and subsequent invocations unless you define them within a different file in the library section.

The first routine limitation has three basic implications:

- First, you cannot use these routines in applications where you repeat the same routine.
- An extension of this limitation is that you cannot invoke two of these routines in a single execution.
- Finally, if you combine any of these routines with other stand-alone routines, you must invoke them as the first stand-alone routine.

The second routine limitation has one basic implication. When using the technique of repeated stacking of the same stand-alone routine, you cannot change certain major fields in the parameter list if the fields are defined in the same file in the library section. However, you can get around this limitation by defining an extra file in the JCL.

First Routine Limitation

If you stack any of the following routines in a single CA-PanAudit Plus execution, they must be the first routine that the CA-PanAudit Plus execution invokes.

```
ADDRCMP          MULTDUP
CAVEVAL          SRCECOMP
CBLCNVRT         STOPORGO
DUPTEST          STRATIF
ENCRYPT          STRTEVL
GAPCHCK          VERSUS
INTERVL
```

Changing Parameter Limitation

The value of the following parameters must not change when you stack the routine repeatedly, unless you define the parameters in different files in the library section:

```
ROUTINE   PARAMETERS
AGING     amount
CAVSAMP   field
DOLUNIT   field
MULTREG   field1
REGSAM    field1  field2
SIMPREG   field1  field2
SPS       field
```

To get around this limitation, define a second ddname (OS) or DLBL file name (DOS) in the JCL that has the same data set name as the original file. This associates two file names in the JCL with the same physical file. Next, define a file in the library section with the file name of the second file, and use the COPY statement to define the identical fields within that file.

When you must change one of the previous parameters in the second invocation statement, specify the second file as the input file and specify the parameter that you want. This uses the same physical file for input in the repeated invocation, but simply refers to it by a different file name. This technique is required only for the parameters listed previously. By defining additional file names, you can use this technique any number of times in a single execution of CA-PanAudit Plus.

If you change any of the previous fields in a repeated stacking application and you do not use this multiple file technique, unpredictable results occur, and no error message is printed. The report may also contain erroneous information.

The following examples demonstrate the technique of stacking stand-alone routines. The examples also show the limitations that this section explained.

Example One — Common Stacking Technique

```
FILE PAYFILE ...
   CLIENT          1    3    N
   GROSS-PAY     23    5    P  2
   ...
FILE SAMP104 ...
FILE SAMP109 ...
FILE SAMP211 ...
*
%SPS1 PAYFILE GROSS-PAY 20000 433281
IF CLIENT NE 104
   GO TO JOB
END-IF
%SPS2 SAMP104
*
%SPS1 PAYFILE GROSS-PAY 25000 433281
IF CLIENT NE 109
   GO TO JOB
END-IF
%SPS2 SAMP109
*
%SPS1 PAYFILE GROSS-PAY 18000 433281
IF CLIENT NE 211
   GO TO JOB
END-IF
%SPS2 SAMP211
```

This example demonstrates a common technique for applying stacking. It involves a payroll file with information from different clients. A CLIENT field, which contains a unique number for each client company, identifies each record in the file.

In this example, SPS (sampling proportional to size) routines create a sample of certain clients in the payroll file. The SPS routines are stacked with the screening code section defining the logic to individually select the clients. Notice that the target parameter is different for each execution. This provides the opportunity to customize the sampling based on historical information for each of the clients. Three separate SPS reports and three separate output files containing the appropriate sample records result from this execution.

Example Two — First Routine Limitation

```
FILE PAYFILE ...
   WEEKLY-GROSS    21    5   P  2
   STATUS-CODE     34    2   N
   ...
%INTERVL1 PAYFILE GROSS-PAY 100 1000 GRAPH 0 2
%INTERVL2
*
%OCCURS1 PAYFILE GRAPH 0 3
%OCCURS2 STATUS-CODE
```

This example demonstrates a stacking technique with two distribution analysis routines in a single execution. Instead of submitting two separate CA-PanAudit Plus jobs, this example performs both the INTERVL and OCCURS routines in a single execution. The first routine performs an interval analysis of the payroll file, then the routine performs an analysis of the occurrence of the STATUS-CODE field.

Notice that INTERVL is the first of the two routines that this example invokes. This is because INTERVL is a stand-alone routine that has the first routine limitation (see First Routine Limitation earlier in this chapter). If you code OCCURS prior to INTERVL, a syntax error occurs.

Example Three — Changing Parameter Technique

```
FILE PAYFILE ...
   GROSS-PAY     23    5    P  2
   NET-PAY       28    5    P  2
   ...
FILE PFILCOP
   COPY PAYFILE
FILE SAMPGRS ...
FILE SAMPNET ...
*
%SPS1 PAYFILE GROSS-PAY 25000 433281
%SPS2 SAMPGRS
*
%SPS1 PFILCOP NET-PAY 20000 433281
%SPS2 SAMPNET
```

In this example, the field parameter is changed in stacked invocations of SPS. The JCL defines the input file twice, once as PAYFILE and once as PFILCOP. These two file names point to the same data set. The library section defines PAYFILE in the same manner as a nonstacked application would normally define PAYFILE. The COPY statement defines the fields in PFILCOP. This provides the same field names for PFILCOP that exist for PAYFILE. The definitions for the two output files follow the COPY statement.

The first SPS routine is invoked with PAYFILE as the input file, GROSS-PAY as the field parameter, and a target of 25,000. The SPS routine is invoked the second time with PFILCOP as the input file, NET-PAY as the field parameter, and a target value of 20,000. Since the value of the field parameter in SPS must not change when the SPS routine is repeatedly stacked, the multiplefile technique is required to change the parameter from GROSS-PAY to NET-PAY. This is accomplished by defining PAYFILE and PFILCOP separately in the library section while defining them as the same file in the JCL. Therefore, SPS obtains the data for the field parameter from the same physical file, while specifying in the invocation statements that the data is from separate files.

Database Use of CA-PanAudit Plus

You can use CA-PanAudit Plus routines to access information in various database structures. The method that you use to access database files depends on whether the routine is an inline or stand-alone routine. If the routine is inline, invoke the CA-PanAudit Plus routine as an integral part of a job activity which is processing the database file. If the routine is stand-alone, CA-PanAudit Plus occasionally requires one additional parameter to inform the routine that it is processing a database file.

In all cases, accessing database files with CA-PanAudit Plus routines requires that you code the CA-Easytrieve Plus statements to properly retrieve the records from the database. It also requires that you code all appropriate checks to assure that RETRIEVE and SELECT statements maintain single complete paths (see the CA-Easytrieve Plus *Reference Guide).* CA-PanAudit Plus routines provide the ability to access information in database files. However, CA-PanAudit Plus does not automate the process of retrieving and constructing records from a database, nor does it check for correct path processing.

Miscellaneous Techniques

The following miscellaneous advanced coding techniques are discussed in this section:

- Creating output files in CA-PanAudit Plus routines
- Suppression of CA-PanAudit Plus macro expansions

Output Files

Many CA-PanAudit Plus routines create an output file (usually in the form of a sample file). In all cases, the library section of the program must contain FILE statements that sufficiently describe the file and the fields necessary to execute the routine.

When coding the attributes for the output file, you must be careful to ensure that the output file has the proper attributes to be able to receive the records written from the input file.

The most common method to ensure that the output file does have the proper attributes is to assign identical attributes to both the input file and the output file. The following example illustrates this method.

Example One — Fixed-length Files with Identical Lengths

```
FILE INFILE  F(100)
*
FILE SAMPFIL  F(100)
*
%RANDPCT1 INFILE 10294 2.0 93584
%RANDPCT2 SAMPFIL
```

Both the input and output file are 100-byte fixed-length files. In most applications you will want to create a sample file that has identical characteristics as the original file.

However, there are applications where you may want to increase the length of a file or change it from variable to fixed-length or conversely. This example and the next one demonstrate how to get these results.

Example Two — Fixed-length Files with Different Lengths

```
FILE INFILE  F(100)
*
FILE SAMPFIL  F(110)
*
%RANDPCT1 INFILE 10294 2.0 93584
%RANDPCT2 SAMPFIL
```

In this example, the records that are selected for sampling are written to an output file that is 10 bytes longer than the input file. The first 100 bytes of each input record are written to the first 100 bytes of the output record. The remaining 10 bytes of the output file have an undetermined value. The output file may have a longer file length than the input file, but it must not have a shorter length.

You could use this method to create space in the sample file to insert additional data such as an audited amount, or a record counter that could be inserted in the screening code section.

Example Three — Fixed-length Files with Different Lengths

```
FILE INFILE  F(100)
*
FILE SAMPFIL  F(105)
ORIGINAL-POSITION    101  5  P  0
*
%RANDPCT1 INFILE 10294 2.0 93584
ORIGINAL-POSITION = RECORD-COUNT(INFILE)
%RANDPCT2 SAMPFIL
```

In this example, the additional five bytes of the sample record indicate the position of the record in the original file. The ORIGINAL-POSITION field is defined in the sample file starting at position 101. The screening code section assigns the RECORD-COUNT of the input file to the ORIGINAL-POSITION field. When a given record is selected for sampling, the first 100 bytes are moved to the output buffer and the final 5 bytes containing the record count do not change. The data in the output buffer is then written to the sample file.

It is important to understand that when you use this technique, CA-PanAudit Plus builds sample records by moving the input data to the output buffer beginning at byte one of the output buffer for the length of the input record. Any data that you insert must be added beyond the length of the input record.

Example Four — Fixed-length to Variable-length Files

```
FILE INFILE  F(100)
*
FILE SAMPFIL  V(104)
*
%RANDPCT1 INFILE 10294 2.0 93584
%RANDPCT2 SAMPFIL
```

This example demonstrates the creation of a variable-length sample file from a 100-byte fixed-length input file. The sample file is defined as 104 bytes to accommodate the four bytes for the record descriptor word. When creating a variable-length sample file from a fixed-length input file, make sure that the length of the sample file is at least four bytes greater than the input file.

Example Five — Variable-length to Fixed-length Files

```
FILE INFILE  V(104)
*
FILE SAMPFIL  F(100)
*
%RANDPCT1 INFILE 10294 2.0 93584
%RANDPCT2 SAMPFIL
```

This example demonstrates the creation of a 100-byte fixed-length sample file from a 104 byte variable-length input file. The sample file is defined as 100 bytes because the 104-byte length of the input file includes four bytes for the record descriptor word, which a fixed-length file does not use. When creating a fixed-length sample file from a variable-length input file, make the minimum length of the sample file four bytes less than the input file.

Output File Summary

When the input file is a fixed-length file, the output file may be:

- Fixed-length with a record length greater than or equal to the record length of the input file
- Variable-length with a record length greater than or equal to the record length of the input file plus four

When the input file is a variable-length file, the output file may be:

- Fixed-length with a record length greater than or equal to the length of the input file minus four
- Variable-length with a record length greater than or equal to the record length of the input file

Suppression of Macro Expansions

To suppress the expansion of the CA-PanAudit Plus statements that comprise the various macros, specify the CA-Easytrieve Plus LIST NOMACROS statement. For example:

```
FILE PAYFILE
   EMPLOYEE-NAME     1    20   A
   NET-PAY          21     5   P  2
   SSN              26     9   N
   HIRE-DATE        35     6   A
*
JOB INPUT PAYFILE
%NUMTEST NET-PAY 'NET-PAY NOT NUMERIC' EMPLOYEE-NAME
*
LIST NOMACROS
*
%DATEVAL HIRE-DATE MMDDYY
IF DATEVAL-FLAG EQ 'NO'
   DISPLAY 'INVALID HIRE DATE FOR ' EMPLOYEE-NAME   +
           '   DATE= ' HIRE-DATE
END-IF
*
LIST MACROS
*
%NUMTEST SSN 'SSN NOT NUMERIC' EMPLOYEE-NAME
```

This example demonstrates the use of the LIST NOMACROS statement to suppress the listing of the CA-PanAudit Plus routines. The first NUMTEST routine expands all macro statements because the default value for the CA-PanAudit Plus system is LIST MACROS. Prior to the DATEVAL routine, the LIST NOMACROS statement suppresses the listing of the DATEVAL macro statements. The only statements that are in the listing for DATEVAL are the five statements listed after the LIST NOMACROS statement. Then, the LIST MACROS statement turns the listing back on prior to the second NUMTEST routine.

You can use the LIST NOMACROS and LIST MACROS statements at any time to turn the macro expansion listings on and off. You can even use these statements in the screening code section to turn the remainder of a listing on or off.

Functional Chart of Inline and Stand-alone Routines

The functional chart of inline and stand-alone routines lists the general and statistical routines. The chart summarizes appropriate functional processing information for each routine and it indicates whether a routine:

- Is inline, stand-alone display, or stand-alone report
- Can be used in a database application
- Contains a FILE statement
- Uses the sample flag technique
- Contains stacking limitations

In the chart, IL indicates an inline routine, SA-D a stand-alone-DISPLAY routines, and SA-R a stand-alone-REPORT routine. YES and NO indicate that a certain feature or limitation applies, and N/A indicates that it is not applicable. The classifications other than Type and Database apply only to stand-alone routines, so all inline routines are listed as N/A. The classifications are summarized after the chart and are discussed in detail earlier in this chapter.

The first column of the chart indicates the type of the routine, which controls the manner in which you can combine CA-Easytrieve Plus LOGIC with CA-PanAudit Plus routines. In the case of stand-alone routines, it specifically controls the coding in the logic-after-invocation section.

The DB column indicates whether you can use a routine in a database application.

The FILE column indicates whether a stand-alone routine defines its own files. This controls the coding of the logic-before-invocation section.

The Sample Flag column indicates whether the stand-alone routine provides a parameter for the sampling flag technique. This provides the ability to identify records that have been selected for sampling.

The stacking limitations column indicates whether a stand-alone routine is limited in its ability to be combined with itself or other stand-alone routines in a single execution of CA-PanAudit Plus.

Functional Chart

Routine	Type	DB	FILE	Flag	Limitation
ADDRCMP	SA-R	YES	YES	NO	YES
AGING	SA-R	YES	NO	NO	YES
ALPHACON	IL	YES	N/A	N/A	N/A
ALPHAGEN	IL	NO	N/A	N/A	N/A
APR	IL	YES	N/A	N/A	N/A
ATTPCT	IL	YES	N/A	N/A	N/A
ATTSAMP	SA-D	YES	NO	YES	NO
BADGEN	IL	NO	N/A	N/A	N/A
CAVEVAL	SA-D	NO	YES	NO	YES
CAVSAMP	SA-D	YES	NO	YES	YES
CBLCNVRT	SA-R	NO	YES	NO	YES
CONVAE	IL	YES	N/A	N/A	N/A
CONVEA	IL	YES	N/A	N/A	N/A
DATECALC	IL	YES	N/A	N/A	N/A
DATECONV	IL	YES	N/A	N/A	N/A
DATEGEN	IL	NO	N/A	N/A	N/A
DATEVAL	IL	YES	N/A	N/A	N/A
DAYSAGO	IL	YES	N/A	N/A	N/A
DAYSCALC	IL	YES	N/A	N/A	N/A
DECRYPT	IL	YES	N/A	N/A	N/A
DISCPCT	IL	YES	N/A	N/A	N/A
DISCSMP	SA-D	YES	NO	YES	NO
DIVIDE	IL	YES	N/A	N/A	N/A
DOLUNIT	SA-D	YES	NO	YES	YES
DUPTEST	SA-R	YES	YES	NO	YES
EACHNTH	IL	YES	N/A	YES	N/A
ENCRYPT	SA-D	YES	YES	NO	YES
EXPO	IL	YES	N/A	N/A	N/A
FILECOMP	SA-D	NO	NO	NO	NO

Routine	Type	DB	FILE	Flag	Limitation
FILEGEN	IL	NO	N/A	N/A	N/A
FLDVALR	SA-R	YES	NO	NO	NO
FLDVALT	SA-R	YES	NO	NO	NO
FLDVALV	SA-R	YES	NO	NO	NO
GAPCHCK	SA-R	YES	YES	NO	YES
GETDATE	IL	YES	N/A	N/A	N/A
INTERVL	SA-R	YES	YES	NO	YES
INTSAMP	SA-D	YES	NO	YES	NO
MULTDUP	SA-R	YES	YES	NO	YES
MULTREG	SA-D	YES	NO	NO	YES
NUMGEN	IL	NO	N/A	N/A	N/A
NUMTEST	IL	YES	N/A	N/A	N/A
OCCURS	SA-R	YES	NO	NO	NO
POPCOUNT	IL	YES	N/A	N/A	N/A
POPSIZE	SA-D	YES	NO	NO	NO
RANDOM	IL	YES	N/A	N/A	N/A
RANDPCT	SA-D	YES	NO	YES	NO
RANDSPAN	IL	YES	N/A	N/A	N/A
RANDXCT	SA-D	YES	NO	YES	NO
REGEVAL	SA-D	NO	NO	NO	NO
REGSAM	SA-D	NO	NO	NO	YES
REGSAMP	SA-D	YES	NO	YES	NO
SIMPREG	SA-D	YES	NO	NO	YES
SPS	SA-D	YES	NO	YES	YES
SQRT	IL	YES	N/A	N/A	N/A
SRCECOMP	SA-R	NO	YES	NO	YES
STDDEV	IL	YES	N/A	N/A	N/A
STOPORGO	SA-D	NO	YES	YES	YES
STRATIF	SA-R	YES	YES	NO	YES
STRTEVL	SA-D	NO	YES	NO	YES

Routine	Type	DB	FILE	Flag	Limitation
TIMECONV	IL	YES	N/A	N/A	N/A
UNBYTE	IL	YES	N/A	N/A	N/A
VARPCT	IL	YES	N/A	N/A	N/A
VARSAMP	SA-D	YES	NO	YES	NO
VERSUS	SA-R	YES	YES	NO	YES
WEEKDAY	IL	YES	N/A	N/A	N/A

Inline Routines (IL)

- Can be combined with other CA-PanAudit Plus Routines O-R CA-Easytrieve Plus logic.
- Are unrestricted or unlimited in usage.

Stand-alone DISPLAY Routines (SAD)

- Use DISPLAY statements to create listings.
- Can accommodate user-coded procedures in the logic-after-invocation section.
- Use special-name REPORT procedures only if you code a REPORT subactivity.

Stand-alone REPORT Routines (SAR)

- Use REPORT subactivities to create listings.
- Cannot accommodate procedures in the logic-after-invocation section.
- Can use special-name REPORT procedures if the routine does not already use them (see Special-name REPORT Procedures earlier in this chapter).

Database Routines (DB)

- Inline routines can be invoked as an integral part of any job activity that is processing a database file. They require no special processing considerations.
- Some stand-alone routines require an additional parameter to inform the routine that it is processing a database file.

- Some stand-alone routines can access database and nondatabase files without the specification of additional parameters.
- Because of the nature of the processing of some stand-alone routines, or because of the operational characteristics that they use, database use of some stand-alone routines is not possible.

FILE Limitations (FILE)

- Routines that define their own files cannot use the logic-before-invocation section.

Sample Flags

- Most routines that create sample files provide the capability to test a reserved field to determine if a record was selected for the sample file.
- The PERFORM procname parameter identifies routines with sample flags.

Stacking Limitations

- Some stand-alone routines can only be the first of any stacked stand-alone routines.
- Some stand-alone routines have limitations regarding the specification of parameters when the routine is repeatedly stacked in a single execution of CA-PanAudit Plus.

Logic-before-invocation Section of Stand-alone Routines

- Located after the library section and before the first CA-PanAudit Plus macro invocation statement.
- Used to preprocess input data, merge or combine input files, or for other preinvocation purposes.
- Routines that define their own files cannot use the logic-before-invocation section. The FILE column in the functional chart of Inline Stand-alone Routines (earlier in this chapter) identifies these routines.

Logic-after-invocation Section of Stand-alone Routines

- Located after the final CA-PanAudit Plus macro invocation statement.
- Use to perform customized processing with the routine and to customize the listing.
- Procedures are performed and REPORTs are printed in the screening code section.

- Procedures and REPORTs are coded in the logic-after-invocation section of stand-alone routines.
- If procedures and REPORTs are used, the procedures must precede the definition of any REPORTs.
- Any special-name REPORT procedures must be coded following the associated REPORT.

Appendix

A Keywords

There are two types of keywords used in the execution of CA-PanAudit Plus:

- CA-Easytrieve Plus
- CA-PanAudit Plus

The CA-Easytrieve Plus keywords are those keywords which are associated with the CA-Easytrieve Plus product and are found in the CA-Easytrieve Plus *Reference Guide*.

The CA-PanAudit Plus keywords are reserved words used by the CA-PanAudit Plus routines.

CA-PanAudit Plus Keywords

CA-PanAudit Plus keywords cannot be used for the following:

- Field names
- File names
- Report names
- Procedure names
- Statement labels

The following list shows the CA-PanAudit Plus keywords and the routines in which they are used:

Keyword	**Routine**
GRAPH	INTERVL,OCCURS,VERSUS
INTERTAB	INTEVL
KEYSB	FILECOMP
LOWERLIM	INTERVL
NOFILE	Most routines with an output file
NOGRAPH	INTERVL,OCCURS,VERSUS
PERCENTAGE	INTERVL
PRIKEYS	FILECOMP
REPORT	DOLUNIT
SECKEYS	FILECOMP
STRATTAB	STRATIF
STRTEVL	STRATIF
SUMMARY	DUPTEST
THRESHOLD	DATECALC,DATECONV

Appendix

B Sample Applications

This appendix presents case studies of simulated auditing projects. For each case study, this chapter provides a description of the auditing environment, separates each auditing project into tasks, and provides CA-PanAudit Plus jobs for each task. The CA-PanAudit Plus jobs execute against a data file produced by a routine which uses the test data generation routines. This chapter also shows output listings from the execution of the jobs.

CA-PanAudit Plus routines that generate the input data and execute the jobs defined by the audit tasks are precoded and exist in the macro library. You can use these CA-PanAudit Plus routines in many ways:

- As an installation validation run stream
- To verify the validity of CA-PanAudit Plus routines
- To create customized versions of the case study data files for testing purposes
- As an instructional facility

You can use the case study routines to create data files that contain information described in the file layout for each case study. The case study jobs can be executed against the data to verify that the results are identical to the results illustrated in this chapter. This will verify that the installation has been properly performed.

Also, you can use the case study routines to verify the validity of the algorithm used by many CA-PanAudit Plus routines. By coding your own jobs to display certain fields in the data files, you can perform desk calculations that verify the validity of the CA-PanAudit Plus routines.

You can customize certain aspects of data file generation through the use of the parameters that the case study system provides. This allows you to create a file with sufficient fields to support the definition of your own sample audit project. You can find details regarding these parameters in the associated case studies.

Finally, you can use the case study system as an instructional facility. This applies to the existing case study jobs that perform predefined tasks and to the customization of an audit project using the case study file generation routines.

Case Study File Generation

Each case study deals with an audit topic. To generate data, case studies use CA-PanAudit Plus test data generation routines and CA-Easytrieve Plus logic. The case study file generation routines generate data for all fields that the case study's file layout defines. You must code the appropriate JCL statements in accordance with the file layout's file attributes.

You can generate case study data files by invoking a routine. Use the following format to invoke the routine:

```
%CSnFILGN ...
```

where n is the number of the case study.

Each file generation routine contains parameters that affect the generation of the data. All parameters are optional, and control the generation and format of dates, or the number of records written to the file.

If you invoke the routine without specifying any parameters, then the default values will generate a file with the current system date as the logical base date for all dates in the file. The default value for the number of records is listed for each case study, and it is the value used in the examples in this chapter. See the appropriate case study for details regarding the parameters.

Case Study Job Execution

After generating a case study file, you can execute CA-PanAudit Plus jobs against the file.

The CA-PanAudit Plus jobs in each case study perform specific tasks. Two methods of organizing jobs are:

- The jobs are logically grouped to perform either related tasks or tasks that can be performed in a single execution of CA-PanAudit Plus.
- To allow you to execute each CA-PanAudit Plus job separately, the jobs exist as single entities.

You can execute case study jobs by invoking a routine. Use the following format to invoke this routine:

```
%CSnJOBx ...
```

where n is the number of the case study, and x identifies the job or group of jobs that you want to execute.

There are groups of case study jobs. A number identifies each group. For example, CS3JOB3 is the third group of jobs for Case Study Three. Letters of the alphabet identify the separately defined case studies. For example, CS1JOBA invokes a routine to perform task A for Case Study One.

Most jobs do not have parameters. However, some jobs have optional parameters that affect the handling of dates. These jobs require the use of the optional parameters only if the associated file generation routine also specified date modification parameters. If you invoke the file generation routine without specifying any parameters, use default values and specify no parameters for the job execution routines that you invoke against that file. See the appropriate case study for details regarding the parameters.

Case Study One

A new staff auditor, with the company for six months, attends a CA-PanAudit Plus training session and is eager to use CA-PanAudit Plus.

The auditor is to perform a survey of the accounts payable function to determine if the payables function warrants a detailed audit. The auditor has heard from conversations with other employees that the operation is a bit lax, and some discounts may have been lost. The auditor decides to perform the following tasks for the initial phase of the survey:

- Determine if any customer has been paid twice for the same invoice.
- Verify that discounts have been taken correctly.

Input

The list that follows provides the layout of the input file for Case Study One:

```
*
*****  FILE AND FIELD DEFINITIONS FOR CASE STUDY ONE
*
FILE ACCTPAY FB (51 5100)
VEN-NUMBER         1   5  N       HEADING ('VENDOR' 'NUMBER')
VEN-NAME           6  12  A       HEADING ('VENDOR' 'NAME')
INV-AMOUNT        18   5  P  2    HEADING ('INVOICE' 'AMOUNT')
INV-NUMBER        23   7  N       HEADING ('INVOICE' 'NUMBER')
INV-DATE          30   6  N       HEADING ('INVOICE' 'DATE')
PMNT-DATE         36   6  N       HEADING ('PAYMENT' 'DATE')
AMT-PAID          42   5  P  2    HEADING ('AMOUNT' 'PAID')
PAID-CODE         47   1  A       HEADING ('PAID' 'CODE')
DISC-PERIOD       48   2  N       HEADING ('DISCOUNT' 'PERIOD')
DISC-PCT          50   2  N       HEADING ('DISCOUNT' 'PERCENT')
```

The Case Study One file generation routine (CS1FILGN) uses the file shown previously as the layout to create data. The file ACCTPAY (accounts payable) contains records with a length of 51 bytes and a block length of 5100 bytes. You must code JCL statements with the appropriate file characteristics when generating or accessing the ACCTPAY file. The macro CS1FILDF contains the previous file and field definitions.

Output

There are no output files for this case study.

Job Number One

This job combines tasks A and B into one execution of CA-PanAudit Plus, as follows:

```
%CS1FILDF
*
*****  TASK A:
*****  CHECK FOR DUPLICATE PAYMENTS OF INVOICES
*
%MULTDUP ACCTPAY U NOFILE VEN-NUMBER INV-NUMBER
IF MULTDUP-FLAG EQ 'YES'
   PRINT DUPLICATE-INVOICES
END-IF
*
REPORT DUPLICATE-INVOICES
CONTROL VEN-NUMBER
TITLE 01 'SPACELY SPROCKET CORPORATION'
TITLE 02 'DUPLICATE ACCOUNTS PAYABLE INVOICES'
LINE 01 VEN-NUMBER INV-NUMBER VEN-NAME INV-AMOUNT
*
*****  TASK B:
*****  CHECK FOR MISSED AND ERRONEOUS DISCOUNTS
*
JOB INPUT ACCTPAY
DEFINE DAYS-BEFORE-PMNT      W   2   N       +
  HEADING ('DAYS BETWEEN' 'INVOICE AND' 'PAYMENT')
DEFINE COR-PMNT-AMT          W   5   P  2     +
  HEADING ('CORRECT' 'PAYMENT' 'AMOUNT')
DEFINE DIFFERENCE            W   5   P  2     +
  HEADING ('OVERPAID')
DEFINE DISC-MISSED           W   5   P  2     +
  HEADING ('DISCOUNT' 'MISSED')
*
IF PAID-CODE NE 'P'
   GO TO JOB
END-IF
%DAYSCALC PMNT-DATE MMDDYY INV-DATE MMDDYY DAYS-BEFORE-PMNT
IF DAYS-BEFORE-PMNT GT DISC-PERIOD
   DISC-MISSED = AMT-PAID * (DISC-PCT / 100)
   PRINT DISCOUNTS-MISSED
ELSE
   COR-PMNT-AMT = INV-AMOUNT - (INV-AMOUNT * (DISC-PCT  / 100))
   IF COR-PMNT-AMT NE AMT-PAID
      DIFFERENCE = AMT-PAID – COR-PMNT-AMT
      PRINT ERRONEOUS-DISCOUNTS
   END-IF
*****  ERRONEOUS DISCOUNTS REPORT
END-IF
*

*****  MISSED DISCOUNTS REPORT
*
REPORT DISCOUNTS-MISSED SPACE 2 SUMSPACE 2
SEQUENCE VEN-NUMBER
CONTROL
TITLE 01 'SPACELY SPROCKET CORPORATION'
TITLE 02 'ACCOUNTS PAYABLE DISCOUNTS MISSED'
LINE 01 VEN-NUMBER VEN-NAME INV-NUMBER INV-DATE PMNT-DATE   +
        DAYS-BEFORE-PMNT DISC-PERIOD DISC-PCT INV-AMOUNT    +
        AMT-PAID DISC-MISSED
*
*
REPORT ERRONEOUS-DISCOUNTS
SEQUENCE VEN-NUMBER
```

```
CONTROL
TITLE 01 'SPACELY SPROCKET CORPORATION'
TITLE 02 'ACCOUNTS PAYABLE ERRONEOUS DISCOUNTS'
LINE 01 VEN-NUMBER VEN-NAME INV-NUMBER INV-DATE INV-AMOUNT   +
        DISC-PERIOD DISC-PCT AMT-PAID DIFFERENCE
LINE 02 POS 4 PMNT-DATE  POS 6 DAYS-BEFORE-PMNT  POS 8 COR-PMNT-AMT
```

First, the routine invokes the CS1FILDF macro, which contains the file and field definitions for the ACCTPAY file. Then task A performs a check for duplicate payments of invoices. The MULTDUP routine checks for duplicate values in two fields, simultaneously. MULTDUP sets the field MULTDUP-FLAG to YES if both the VEN-NUMBER and INV-NUMBER fields are identical in more than one record in the ACCTPAY file. When the MULTDUP-FLAG equals YES, task A prints a line of the DUPLICATE-INVOICES report.

Task B checks for any missing or erroneous discounts. First, task B defines four working storage fields which it uses in subsequent calculations. Next, if the PAID-CODE does not indicate that the invoice has been paid, then the GO TO JOB statement bypasses the record.

Task B then uses DAYSCALC to calculate the number of days between the date of payment and the invoice date. If this value is greater than the discount period for this invoice, the discount was missed, and task B prints a report line. Otherwise, task B calculates the correct discount amount and compares the discount amount with the actual amount paid for the invoice. If the values are not equal, the discount was incorrectly calculated, and task B prints a line of the ERRONEOUS-DISCOUNTS report.

Output - Task A

```
            SPACELY SPROCKET CORPORATION                        PAGE      1
         DUPLICATE ACCOUNTS PAYABLE INVOICES

   VENDOR   INVOICE       VENDOR          INVOICE
   NUMBER   NUMBER         NAME            AMOUNT

   00131    0006036    VENDOR131           40,900.60
            0006036    VENDOR131           40,900.60
   00131                                   81,801.20

                                           81,801.20
```

The report shows the duplicate vendor and invoice numbers. The report also lists the vendor names and amounts of the invoices. The VEN-NUMBER field controls the report, so whenever the value of that field changes, task A prints a subtotal.

Output - Task B - Discounts Missed

```
                         SPACELY SPROCKET CORPORATION                          PAGE      1
                       ACCOUNTS PAYABLE DISCOUNTS MISSED

                                        DAYS BETWEEN
VENDOR      VENDOR  INVOICE  INVOICE  PAYMENT  INVOICE AND  DISCOUNT  DISCOUNT  INVOICE     AMOUNT      DISCOUNT
NUMBER        NAME  NUMBER    DATE     DATE      PAYMENT     PERIOD   PERCENT   AMOUNT       PAID       MISSED

00101   VENDOR101  0005604  112789  011390       47           45         01     30,817.20   30,817.20    308.17
00109   VENDOR109  0005612  120489  011790       44           30         01     38,729.20   38,729.20    387.29
```

00110	VENDOR110	0006015	012190	021090	20	15	05	3,305.83	3,305.83	165.29
00114	VENDOR114	0005617	011290	022690	45	30	03	3,995.68	3,995.68	119.87
00121	VENDOR121	0005423	120889	022890	82	60	02	13,901.63	13,901.63	278.03
00124	VENDOR124	0005828	120889	021190	65	45	01	63,631.15	63,631.15	636.31
00125	VENDOR125	0005025	011090	021690	37	15	01	39,169.71	39,169.71	391.69
00131	VENDOR131	0005433	122689	022890	64	60	01	45,796.86	45,796.86	457.96
00131	VENDOR131	0006237	122989	021690	49	30	01	30,873.16	30,873.16	308.73
00135	VENDOR135	0005437	111389	011090	58	30	03	41,553.50	41,553.50	1,246.60
00140	VENDOR140	0005844	011090	022790	48	45	01	38,150.53	38,150.53	381.50
00147	VENDOR147	0005047	120489	011590	42	30	02	44,650.25	44,650.25	893.00
00147	VENDOR147	0005851	010290	021390	42	15	02	8,517.94	8,517.94	170.35
00148	VENDOR148	0005852	122589	021990	56	30	01	10,635.39	10,635.39	106.35
00149	VENDOR149	0005853	110889	020490	88	60	01	49,276.07	49,276.07	492.76
00163	VENDOR163	0006068	010390	022590	53	45	01	38,925.33	38,925.33	389.25
00166	VENDOR166	0005066	010690	021490	39	30	02	54,266.51	54,266.51	1,085.33
00170	VENDOR170	0005271	010590	020590	31	15	05	2,014.03	2,014.03	100.70
00179	VENDOR179	0005883	102889	010290	66	60	05	10,717.41	10,717.41	535.87
00179	VENDOR179	0006084	121689	022890	74	45	01	18,467.38	18,467.38	184.67
00184	VENDOR184	0005285	122389	012390	31	15	02	26,005.39	26,005.39	520.10
00191	VENDOR191	0006096	011590	021590	31	30	01	33,943.36	33,943.36	339.43
00222	VENDOR222	0006127	011990	022690	38	30	02	48,909.87	48,909.87	978.19
00225	VENDOR225	0005125	121989	012190	33	15	03	50,574.15	50,574.15	1,517.22
00227	VENDOR227	0005931	012490	022390	30	15	05	25,571.82	25,571.82	1,278.59
00228	VENDOR228	0005731	121189	010690	26	15	02	44,645.64	44,645.64	892.91
00231	VENDOR231	0005131	112989	011190	43	15	02	57,372.31	57,372.31	1,147.44
00233	VENDOR233	0005736	121389	020390	52	30	01	54,619.05	54,619.05	546.19
00254	VENDOR254	0005154	121789	022390	68	45	01	6,019.37	6,019.37	60.19
00266	VENDOR266	0005769	122489	020490	42	15	01	64,510.26	64,510.26	645.10
00269	VENDOR269	0005169	122789	020490	39	15	01	37,394.67	37,394.67	373.94
00271	VENDOR271	0005975	010790	021790	41	15	02	62,473.74	62,473.74	1,249.47
00272	VENDOR272	0005775	121589	010790	23	15	05	14,221.08	14,221.08	711.05
00279	VENDOR279	0005983	111589	012390	69	60	02	43,861.04	43,861.04	877.22
00283	VENDOR283	0005183	122089	022690	68	60	02	16,587.52	16,587.52	331.75
00284	VENDOR284	0005586	010890	022490	47	30	01	4,249.53	4,249.53	42.49
00290	VENDOR290	0005793	101789	010190	76	60	01	31,822.39	31,822.39	318.22
00297	VENDOR297	0005599	010890	020490	27	15	02	7,441.54	7,441.54	148.83
								1,217,617.49	1,217,617.49	20,618.05

The DISCOUNTS-MISSED report shows the invoices which, if paid before the discount period, could have taken discounts. The DAYS BETWEEN INVOICE AND PAYMENT column lists the number of days before payment. The DISCOUNT PERIOD column lists the number of days in which the customer must make the payment To receive a discount. The report also lists the amount paid and the amount of discount missed.

Output - Task B - Erroneous Discounts

SPACELY SPROCKET CORPORATION PAGE 1
ACCOUNTS PAYABLE ERRONEOUS DISCOUNTS

VENDOR NUMBER	VENDOR NAME	INVOICE NUMBER	INVOICE DATE	INVOICE AMOUNT	DISCOUNT PERIOD	DISCOUNT PERCENT	AMOUNT PAID	OVERPAID
00105	VENDOR105	0005809	012790	72,688.09	15	03	66,982.06	3,525.38-
			020890		12		70,507.44	
00113	VENDOR113	0005817	121489	3,094.79	30	01	3,191.50	127.66
			010990		26		3,063.84	
00114	VENDOR114	0006019	011790	31,190.94	15	02	29,038.76	1,528.36-
			012990		12		30,567.12	
00124	VENDOR124	0005426	011390	65,296.36	30	02	65,969.51	1,979.08
			021190		29		63,990.43	
00138	VENDOR138	0005842	012090	11,014.99	30	02	10,578.79	215.90-
			021690		27		10,794.69	

00152	VENDOR152	0005253	122689	70,659.02	60	01	66,454.79	3,497.63-
			022290		58		69,952.42	
00159	VENDOR159	0005863	112289	38,870.71	60	02	39,680.51	1,587.22
			011890		57		38,093.29	
00171	VENDOR171	0005674	122589	25,099.81	45	02	25,892.43	1,294.62
			020490		41		24,597.81	
00173	VENDOR173	0005877	121189	48,162.13	30	01	45,296.47	2,384.03-
			010790		27		47,680.50	
00196	VENDOR196	0006101	011990	7,508.72	15	02	7,586.12	227.58
			020190		13		7,358.54	
00211	VENDOR211	0005714	010690	37,842.39	45	02	37,842.38	756.84
			021690		41		37,085.54	
00224	VENDOR224	0005727	011990	52,088.47	30	05	47,504.67	1,979.37-
			021690		28		49,484.04	
00232	VENDOR232	0005735	010590	11,483.83	15	02	11,846.47	592.32
			011990		14		11,254.15	
00283	VENDOR283	0005987	120789	8,036.63	30	02	8,290.41	414.52
			010190		25		7,875.89	
00288	VENDOR288	0006193	111089	73,072.81	60	03	69,463.00	1,417.62-
			010890		59		70,880.62	
00296	VENDOR296	0006201	012690	37,196.53	15	01	38,762.69	1,938.13
			020790		12		36,824.56	
				593,306.22			574,380.56	5,630.32-

The ERRONEOUS-DISCOUNTS report shows the discounts that were incorrectly calculated. The report lists the invoice amount, discount percentage, amount paid, and the amount that was over or underpaid. The ERRONEOUS-DISCOUNTS report calculates totals for the appropriate columns.

Job Summary

The following list provides a summary of the Case Study One CA-PanAudit Plus routines and their associated parameters and files:

ROUTINE	PURPOSE	PARAMETERS	FILES
CS1FILDF	file definition	none	ACCTPAY
CS1FILGN	generate file	DATEVALUE, FORMAT, NUM-RECS	ACCTPAY
CS1JOB1	tasks A and B	none	ACCTPAY
CS1JOBA	task A	none	ACCTPAY
CS1JOBB	task B	none	ACCTPAY

Syntax

```
%CS1FILGN [DATEVALUE date] [FORMAT value] [NUM-RECS number]
```

`[DATEVALUE date]`

DATEVALUE controls the generation of dates. For all paid invoices, the invoice and payment dates must be in the past. DATEVALUE defines the latest date for payment. The associated invoice date will be before the payment date. The default value is the current system date.

`[FORMAT value]`

Specify the format of DATEVALUE. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
```

Note: The only valid Julian format is YYDDD.

The default value for FORMAT is MMDDYY. In most cases, you only need to specify FORMAT if you specify DATEVALUE. However, if you allow DATEVALUE to default, and the current system date is not in the format MMDDYY, you must specify the proper format of the current system date.

`[NUM-RECS number]`

Specifies the number of records that CS1FILGN is to generate. The default value is 1243.

Operation

Allowing all three parameters in CS1FILGN to default produces results identical to those shown in the Case Study One reports, with the exception of the exact values for the dates. The relationships between dates remain the same, as do all other numeric results. The generated dates are based on the date that you specify for DATEVALUE. If DATEVALUE is allowed to default, the generated dates will be based on the current system date. Specify a DATEVALUE of 07/19/85 to produce the exact same results, including the actual dates as shown in the Case Study One reports.

CS1JOB1 combines both tasks A and B. You must perform Task A first, since MULTDUP has the FILE limitation and cannot have logic before the invocation. CS1JOBA and CS1JOBB separately perform tasks A and B. For information on the FILE limitation, see the chapter "Advanced Techniques."

The ACCTPAY file contains records 51 bytes long with a block length of 5100 bytes. CS1FILGN writes the generated records to the ACCTPAY file, while CS1JOB1, CS1JOBA, and CS1JOBB use ACCTPAY as an input file.

Case Study Two

Operating management requests the auditing department to look at receivables management and to provide a quick overview of the extent and quality of receivables. The audit manager decides to perform the following tasks To satisfy those requirements:

- Foot the receivables inventory and reconcile it to the general ledger.
- Test the receivables document file to ensure that documents obligating the customer to pay are present.
- Age the receivables by customer.

The audit manager passes these requirements to one of the staff auditors. The auditor quickly sees that he can obtain most of the necessary information in a single execution of CA-PanAudit Plus. An AGING analysis satisfies requests one and three. An attributes sampling routine satisfies request two by creating a statistically valid sample file for the verification of documents.

Input

The following list is a file layout of the input file for Case Study Two:

```
*
*****  FILE AND FIELD DEFINITIONS FOR CASE STUDY TWO
*
FILE ACCTREC FB (31 3100)
CUST-NR           1   5  N      HEADING ('CUSTOMER' 'NUMBER')
ACCT-STATUS       6   1  A      HEADING ('ACCOUNT' 'STATUS')
INV-NR            7   7  N      HEADING ('INVOICE' 'NUMBER')
INV-DATE         15   6  N      HEADING ('INVOICE' 'DATE')
AMOUNT-DUE       21   5  P  2   HEADING ('AMOUNT' 'DUE')
DUE-DATE         26   6  N      HEADING ('DUE' 'DATE')
```

The Case Study Two file generation routine (CS2FILGN) uses the previous file layout to create data. The file ACCTREC (accounts receivable) contains records 31 bytes long with a block length of 3100 bytes. You must code JCL statements with the appropriate file characteristics when generating or accessing the ACCTPAY file. The macro CS2FILDF contains the previous file and field definitions.

Output

Task B performs attribute sampling and generates an output file. This file is named ARSAMP and contains fixed-length records 31 bytes long.

Job Number One

This job combines tasks A and B into one execution of CA-PanAudit Plus, as follows:

```
%CS2FILDF
*
*****  DEFINE OUTPUT FILE FOR ATTRIBUTES SAMPLING
*
FILE ARSAMP F (31)
   COPY ACCTREC
*
*****  TASK A:
*****  PERFORM AGING
*
%AGING1 ACCTREC DUE-DATE YYMMDD AMOUNT-DUE DAYS OVERDUE
   IF ACCT-STATUS =  'C'
      GO TO JOB
   END-IF
%AGING2 CUST-NR CURANGE REPORTYPE SUMMARY
*
*****  OBTAIN EXACT POPULATION COUNT FOR ATTRIBUTES SAMPLING
*
JOB INPUT ACCTREC
DEFINE NUM-RECS    S   4   P   0   VALUE (0)
IF ACCT-STATUS NE 'C'
   NUM-RECS = NUM-RECS + 1
END-IF
*
*****  TASK B:
*****  PERFORM ATTRIBUTES SAMPLING
*
%ATTSAMP1 ACCTREC NUM-RECS 95 3.0 3.0 987531
   IF ACCT-STATUS =  'C'
      GO TO JOB
   END-IF
%ATTSAMP2 ARSAMP
*
*****  PRINT ATTRIBUTES SAMPLE POPULATION
*
JOB INPUT ARSAMP
PRINT ATT-SAMPLE
*
*****  ATTRIBUTES SAMPLE REPORT
*
REPORT ATT-SAMPLE
CONTROL
TITLE 01 'COGSWELL COG CORPORATION'
TITLE 02 'ATTRIBUTES SAMPLE OF RECEIVABLES FILE'
LINE 01 CUST-NR INV-NR INV-DATE AMOUNT-DUE DUE-DATE
```

First, the routine invokes the CS2FILDF macro, which contains the file and field definitions for the ACCTREC file. The file definition for ARSAMP follows the invocation of the CS2FILDF macro. ARSAMP is a sample file that task B generates. The COPY ACCTREC statement specifies that the field names, lengths, and attributes of the ARSAMP file are identical to those in the ACCTREC file.

Task A performs the aging analysis. The GO TO JOB statement bypasses all records with an account status of C (closed). This eliminates from processing all records that have been paid and are waiting to be deleted from the receivables file.

The next section of code is an example of logic before the invocation of a stand-alone routine. The subsequent ATTSAMP routine requires an exact population count as a parameter. To obtain an exact count, an initial job is coded that accumulates the exact count in the NUM-RECS field. This job bypasses records with an account status of C and increments a counter for all other records. The value calculated for NUM-RECS is specified as the size parameter in the subsequent ATTSAMP routine. This allows ATTSAMP to execute with a dynamically calculated size parameter and assures that the correct number of records are written to the sample file.

The ATTSAMP routine specifies a 95 percent confidence, 3 percent precision, 3 percent error rate, and the dynamically calculated value for population size. The perform attributes sampling job of task B then writes the appropriate number of records to the ARSAMP file. The print attributes sample population job prints a report of the items in the sample file. Items listed in the report will be checked for proper documentation.

Output - Task A

```
                                         CA-PANAUDIT PLUS AGING REPORT                           PAGE      1

                      FILE NAME: ACCTREC    DATE FIELD: DUE-DATE    AMOUNT FIELD: AMOUNT-DUE

                                         RANGE 1       RANGE 2      RANGE 3

                                          1  -  30     31  -  60    OVER    60
                                         DAYS OVERDUE  DAYS OVERDUE DAYS OVERDUE

                                           AGING OF ACCOUNTS AS OF  3/01/90

CUSTOMER
 NUMBER        CURRENT            RANGE 1            RANGE 2            RANGE 3               TOTAL

00100        139,367.28               .00          42,980.14                .00          182,347.42
00101        158,835.10            419.14          32,598.87                .00          191,853.11
00102         90,269.82         71,734.35                .00          37,489.73          199,493.90
00103        131,473.85               .00                .00                .00          131,473.85
00104        120,054.23               .00          13,019.36          36,328.99          169,402.58
00105        111,086.32               .00                .00          16,192.05          127,278.37
00106        115,189.35               .00          20,020.27                .00          135,209.62
00107        122,646.45               .00          20,054.97                .00          142,701.42
00108        140,454.63               .00                .00         117,430.93          257,885.56
00109        214,744.32               .00                .00                .00          214,744.32
00110         83,362.94         19,585.20                .00                .00          102,948.14
00111         53,758.41         30,494.07                .00                .00           84,252.48
00112        165,287.82               .00                .00                .00          165,287.82
00113        194,012.42         31,388.69                .00                .00          225,401.11
00114        127,543.15               .00          54,785.91                .00          182,329.06
00115         22,960.35         87,950.42                .00                .00          110,910.77
00116        131,235.49         24,816.81          26,636.34                .00          182,688.64
00117        109,583.67               .00                .00                .00          109,583.67
00118        103,218.09          7,310.55           2,438.01                .00          112,966.65
   .             .                 .               .              .                     .
   .             .                 .               .              .                     .
   .             .                 .               .              .                     .
01221         79,288.13         26,761.15                .00                .00          106,049.28
01222        117,406.66         52,241.17                .00                .00          169,647.83
01223        126,867.59               .00                .00          38,501.23          165,368.82
01224        238,756.29          2,824.42          16,330.22          39,115.70          297,026.63
01225        197,681.70         48,734.22          29,475.07           8,022.05          283,913.04
```

01226	167,261.51	.00	.00	.00	167,261.51
01227	101,906.14	33,898.50	.00	16,885.20	152,689.84
01228	59,184.47	.00	.00	.00	59,184.47
01229	137,547.02	16,277.64	.00	8,659.79	162,484.45
01230	131,856.18	.00	.00	.00	131,856.18
01231	191,316.60	34,909.24	.00	.00	226,225.84
01232	67,795.20	.00	.00	.00	67,795.20
01233	119,902.91	74,815.42	.00	6,908.54	201,626.87
01234	86,611.18	10,247.32	.00	.00	96,858.50
01235	160,122.05	44,184.44	.00	30,980.37	235,286.86
01236	191,861.56	.00	.00	12,052.78	203,914.34
	157,653,062.32	22,597,269.94	8,716,730.37	14,076,935.29	203,043,997.92

NUMBER OF RECORDS SELECTED FOR THIS REPORT: 8,101

The AGING analysis report lists the current accounts (accounts which are not overdue) and three ranges of 30 days each. The REPORTYPE is specified as SUMMARY, which consolidates all receivables for a given customer and summarizes them into one line item.

Output - Task - B Attribute Sampling Report

```
            ATTRIBUTE SAMPLING REPORT
                INPUT PARAMETERS

INPUT FILENAME                          ACCTREC
TOTAL POPULATION SIZE                     8,101
REQUIRED PRECISION                           3.00
REQUIRED CONFIDENCE LEVEL                  95
ERROR RATE                                   3.00

              SAMPLE RESULTS

SAMPLE PERCENTAGE REQUIRED                   1.50598691%
SAMPLE SIZE REQUIRED                      122

              SAMPLE FILE

NUMBER OF RECORDS PROCESSED               8,101
NUMBER OF RECORDS REQUESTED                 122
NUMBER OF RECORDS IN SAMPLE FILE            122

FILE ARSAMP WILL BE CREATED
```

This report shows that 8,101 records were processed in the ACCTREC file. The calculated sample size was 1.50598691 percent and resulted in a sample file of 122 records. The report also confirms that 122 records were written to the sample file.

Output - Task B - Sample File Listing

COGSWELL COG CORPORATION   PAGE 1

ATTRIBUTES SAMPLE OF RECEIVABLES FILE

CUSTOMER NUMBER	INVOICE NUMBER	INVOICE DATE	AMOUNT DUE	DUE DATE
00223	0001028	900205	39,171.86	900307
00633	0001109	900211	17,854.96	900313
00988	0001254	900204	17,392.79	900306
01224	0001357	900208	16,834.46	900310
00255	0001382	900101	47,417.43	900131

```
00510     0001395   900203         40,707.11    900305
00738     0001435   900214         23,461.48    900316
00957     0001451   900216         39,046.54    900318
00578     0001503   900131         32,333.20    900302
00718     0001518   900201         24,001.42    900303
00604     0001564   900209         26,537.89    900311
00903     0002022   900212         48,794.06    900314
00688     0002107   900226         12,770.62    900328
00913     0002132   900213         15,019.57    900315
00556     0002136   900224          5,108.74    900326
00257     0002316   900218         31,078.94    900320
00990     0002445   900218         15,335.03    900320
00573     0002472   900218          7,876.30    900320
00580     0002499   900228         48,887.70    900330
00391     0002564   891222         49,752.76    900121
00169     0002566   900206         26,775.40    900308
00955     0002580   900212         20,862.65    900314
00328     0002596   900225          9,821.31    900327
01008     0002627   900218         24,456.77    900320
00230     0002650   900211         24,096.79    900313
00622     0002682   900105         45,030.19    900204
00122     0002757   900218         26,444.53    900320
  .        .     .              .        .
  .        .     .              .        .
  .        .     .              .        .
00426     0008363   900104         42,967.46    900203
00323     0008407   900228          9,001.28    900330
00628     0008428   900205         17,886.81    900307
01078     0008467   900225         37,835.46    900327
00988     0008471   900222          6,299.38    900324
00186     0008489   900213         31,945.40    900315
00325     0008498   900118         18,778.41    900217
01015     0008593   900217         44,631.44    900319
01025     0008685   900228          1,452.94    900330
00460     0008777   900206          8,329.20    900308
00316     0008779   900228         45,188.98    900330
00797     0008890   891208            987.91    900107
01059     0009117   900228          6,449.78    900330
00423     0009167   900207          3,337.44    900309
00375     0009479   900213          8,687.22    900315
00547     0009649   900208         22,613.56    900310
01205     0009682   900121         42,485.68    900220
00560     0009700   900226         28,353.67    900328
00669     0009719   900205         31,494.34    900307
01076     0009789   900220         22,698.66    900322
00645     0009820   900228         23,303.64    900330
00574     0009892   900202         46,599.69    900304
```

This report lists various fields from all records in the sample file. The previous records will be checked for proper receivables documentation.

Job Summary

The following is a summary of the Case Study Two CA-PanAudit Plus routines and their associated parameters and files:

```
ROUTINE   PURPOSE          PARAMETERS                  FILES
CS2FILDF  file definition  none                        ACCTREC
CS2FILGN  generate file    DATEVALUE, FORMAT, NUM-RECS  ACCTREC
CS2JOB1   tasks A and B    DATEVALUE, FORMAT           ACCTREC
CS2JOBA   task A           DATEVALUE, FORMAT           ACCTREC
CS2JOBB   task B           none                        ACCTREC, ARSAMP
```

Syntax

```
%CS2FILGN [DATEVALUE date] [FORMAT value] [NUM-RECS number]
```

`[DATEVALUE date]`

DATEVALUE controls the generation of dates. For all unpaid receivables, the invoice and due dates must be in the past. DATEVALUE defines a date from which invoice dates are calculated. Appropriate due dates are calculated from the invoice date. The default value is the current system date.

`[FORMAT value]`

Specify the format of DATEVALUE. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
```

Note: The only valid Julian format is YYDDD.

The default value for FORMAT is MMDDYY. In most cases, you only need to specify FORMAT if you specify DATEVALUE. However, if you allow DATEVALUE to default, and the current system date is not in the format MMDDYY, you must specify the proper format of the current system date.

`[NUM-RECS number]`

Specifies the number of records that CS2FILGN is to generate. The default value is 8954.

Syntax

```
%CS2JOB1 [DATEVALUE date] [FORMAT value]

                    or

%CS2JOBA [DATEVALUE date] [FORMAT value]
```

`[DATEVALUE date]`

DATEVALUE controls the BASEDATE of the AGING routine. The default value is the current system date.

```
[FORMAT value]
```

Specify the format of DATEVALUE. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
```

The default value for FORMAT is MMDDYY. In most cases, you only need to specify FORMAT if you specify DATEVALUE. However, if you allow DATEVALUE to default, and the current system date is not in the format MMDDYY, you must specify the proper format of the current system date.

Operation

Allowing all three parameters in CS2FILGN to default produces results identical to those shown in the Case Study Two reports, with the exception of the exact values for the dates. The relationships between dates remain the same, as do all other numeric results. The generated dates are based on the date that you specify for DATEVALUE. If DATEVALUE is allowed to default, the generated dates will be based on the current system date. Specify a DATEVALUE of 07/19/85 to produce the exact same results, including the actual dates, as shown in the Case Study Two reports.

In most cases, if you allow DATEVALUE to default when the file is generated, you can also allow DATEVALUE to default when jobs CS2JOB1 or CS2JOBA are run. However, if the file was generated on a previous day, jobs CS2JOB1 or CS2JOBA invoke AGING using the current system date for the BASEDATE. This creates a valid aging of the file from the current system date, but does not reproduce the same results as on previous days, because the BASEDATE for the analysis is defaulting to the current system date. This causes the age of a record to change with each subsequent day that you execute the routine. To maintain a consistent report, specify DATEVALUE as the date that the file was generated.

For the same reasons, to produce an AGING report identical to the previous listings, specify a DATEVALUE of 07/19/85 not only in CS2FILGN but also in CS2JOB1 and CS2JOBA.

The ACCTREC file contains records 31 bytes long with a block length of 3100 bytes. The ARSAMP file contains fixed-length records that are 31 bytes long. CS2FILGN writes the generated records to the ACCTREC file, while CS2JOB1, CS2JOBA, and CS2JOBB use ACCTREC as an input file. In addition, CS2JOB1 and CS2JOBB create the sample file, ARSAMP.

Case Study Three

The organization being audited is a state-wide bank holding company. Its data processing is centralized at one location, with terminals and remote printers at various offices. The audit organization itself is, for the most part, centralized. The largest group of the financial auditors are at the main site, but some resident financial auditors are located at a few of the larger member banks. EDP audit support is provided at the main site.

CA-PanAudit Plus is installed at the main site, and its facilities are available to all auditors at any location. Major file/data definitions have been created and cataloged in the data dictionary by the EDP auditors. All financial auditors have been given standard training in the use of CA-PanAudit Plus and are encouraged to use it. EDP auditors provide whatever help is necessary, as well as assist in implementing more technically complex tasks.

The resident auditor at one of the larger banks (not the central site) prepares to do an installment loan audit. He is fairly new and has not been through the installment loan department yet. He reads the previous audit reports and sketches out his work program. He also wants to have an idea of what the installment loan population looks like. He decides to use CA-PanAudit Plus to get a frequency distribution in $2500 intervals.

The coding to provide the frequency distribution report is as follows:

```
%CS3FILDF
*
%INTERVL1 ILMAST LOAN-BAL 2500 50000 GRAPH 0 2
    IF STAT-CODE = 9 10
       GO TO JOB
    END-IF
%INTERVL2
```

The %CS3FILDF statement invokes the data dictionary member for the installment loan master file. This statement will be used to invoke the file and field definitions for the installment loan file for all jobs in this case study.

The invocation of INTERVL1 is followed by statements which bypass status codes nine and ten because they represent closed accounts which are to be purged from the master file. This logic will be employed in most routines for this case study. The parameters in the INTERVL1 invocation specify intervals of $2500 and a materiality of $50,000. Materiality is an amount above which all records are placed in a separate stratum for possible special review.

The output generated by this job has two reports:

- A frequency analysis of loan balances by interval
- A frequency analysis graph of loan balances

```
7/19/89                 FREQUENCY ANALYSIS OF LOAN-BAL                          PAGE  1
                          FROM INPUT FILE ILMAST
                   WITH AN INTERVAL OF        2,5000.00   AND A MATERIALITY OF        50,000.00
                          MAXIMUM VALUE:     64,8000.00  MINIMUM VALUE:          .00

 --------RANGE-----------              TOTAL            COUNT       PCT        MEAN        STD DEV

          <      0.00                    .00               0        .0          .00           .00
          =      0.00                    .00               2        .0          .00           .00
    0.01 -    2500.00             366,311.13             284       1.1     1,289.83        681.88
 2500.01 -    5000.00             849,743.78             226       2.5     3,759.93        699.06
 5000.01 -    7500.00             976,710.06             160       2.9     6,104.44        713.39
 7500.01 -   10000.00           1,287,478.44             149       3.9     8,640.79        712.77
10000.01 -   12500.00           1,382.313.54             123       4.1    11,238.32        780.60
12500.01 -   15000.00           1,781,361.72             130       5.3    13,702.78        709.09
15000.01 -   17500.00           1,686,971.75             104       5.0    16,220.88        738.83
17500.01 -   20000.00           1,973,500.01             105       5.9    18,795.24        722.94
20000.01 -   22500.00           2,038,888.20              96       6.1    21,238.42        750.81
22500.01 -   25000.00           1,654,930.46              70       5.0    23,641.86        660.42
25000.01 -   27500.00           1,862,243.13              71       5.6    26,228.78        693.29
27500.01 -   30000.00           2,004,412.14              70       6.0    28,634.46        749.79
30000.01 -   32500.00           1,994,340.55              64       6.0    31,161.57        697.97
32500.01 -   35000.00           1,550,652.44              46       4.6    33,709.84        781.85
35000.01 -   37500.00           1,886,951.85              52       5.6    36,287.54        719.43
37500.01 -   40000.00           1,818,469.68              47       5.4    38,690.84        737.49
40000.01 -   42500.00           1,614,703.05              39       4.8    41,402.64        694.13
42500.01 -   45000.00             966,573.63              22       2.9    43,935.17        602.97
45000.01 -   47500.00           1,339,677.17              29       4.0    46,195.76        673.24
47500.01 -   50000.00             886,787.03              18       2.7    49,265.95        594.28
          >  50000.00           3,503,127.19              63      10.5    55,605.19      3,756.94

FINAL TOTALS                   33,426,146.95           1,970      99.9    16,967.59     14,452.04
POSITIVE TOTAL                 33,426,146.95
NEGATIVE TOTAL                           .00
ABSOLUTE VALUE TOTAL           33,426,146.95
```

The output shows each interval of $2500 and the respective totals, frequency, and percentage.

The following output is a graphic representation of the percentages where each asterisk in the graph represents one percent:

```
7/19/89                          FREQUENCY ANALYSIS GRAPH OF LOAN-BAL
                                      FROM INPUT FILE ILMAST
        WITH AN INTERVAL OF          2,500.00   AND A MATERIALITY OF            50,000.00
                       MAXIMUM VALUE:      64,800.00   MINIMUM VALUE:                .00


------------ RANGE -----------       PCT

             <       0.00            .0
             =       0.00            .0
        0.01 -    2500.00           1.1          **
     2500.01 -    5000.00           2.5          *****
     5000.01 -    7500.00           2.9          ******
     7500.01 -   10000.00           3.9          ********
    10000.01 -   12500.00           4.1          ********
    12500.01 -   15000.00           5.3          ***********
    15000.01 -   17500.00           5.0          **********
    17500.01 -   20000.00           5.9          ************
    20000.01 -   22500.00           6.1          ************
    22500.01 -   25000.00           5.0          **********
    25000.01 -   27500.00           5.6          ***********
    27500.01 -   30000.00           6.0          ************
```

```
30000.01 -   32500.00         6.0          ************
32500.01 -   35000.00         4.6          *********
35000.01 -   37500.00         5.6          ***********
37500.01 -   40000.00         5.4          ***********
40000.01 -   42500.00         4.8          **********
42500.01 -   45000.00         2.9          ******
45000.01 -   47500.00         4.0          ********
47500.01 -   50000.00         2.7          *****
         >   50000.00        10.5          *********************
```

The auditor now has a good idea of the distribution of loan balances in the file and will use this report to assist him in making intelligent decisions for parameter values in the following jobs.

Audit Task

After reviewing the frequency analysis, the auditor completes his work program and determines where information from computer files may be of use. He identifies the following tasks:

A. Age the loans to test delinquency.

B. Look at extensions by loan officer.

C. Ensure that loans fall into proper categories.

D. Test to determine that first payment is within 45 days of the loan date.

E. Ensure that the loan is not past maturity date with a positive balance.

F. Recompute earned/unearned interest to ensure that the right amount was taken into earnings.

G. Take an attribute sample and examine loan files to ensure that all of the proper documents (note, collateral assignment, and so on) are present.

H. Take a stratified monetary sample and send out positive confirmations.

Examining his work program, the auditor is reasonably sure that he can accomplish tasks A through E himself. He is not so sure about the last three tasks (F, G, and H) and asks the EDP auditor to do those.

Input

```
FILE ILMAST FB(149 14900)
LOAN-NBR          1   8  N     HEADING ('LOAN' 'NUMBER')
ACCT-NBR          9   8  N     HEADING ('ACCOUNT' 'NUMBER')
NAME             17  12  A     HEADING ('BORROWER NAME')
STAT-CODE        29   2  N     HEADING ('ACCT' 'STAT' 'CODE')
TYPE-CODE        31   2  N     HEADING ('ACCT' 'TYPE' 'CODE')
LOAN-OFCR        33   3  N     HEADING ('LOAN' 'OFCR')
ORIG-BAL         39   4  P  2  HEADING ('ORIGINAL' 'BALANCE')
DUE-DATE         45   6  N                          +
                 HEADING ('DATE NEXT' 'PAYMENT') +
                 MASK    (A  BWZ  'Z9/99/99')
LOAN-BAL         51   4  P  2  HEADING ('LOAN' 'BALANCE')
LOAN-DATE        57   6  N     HEADING ('LOAN ISSUE' 'DATE') +
                               MASK    (A  BWZ)
MATUR-DATE       63   6  N     HEADING ('MATURITY' 'DATE') +
```

```
                              MASK    (A  BWZ)
DT-LAST-PAY      69   6  N                      +
                 HEADING ('DATE LAST' 'PAYMENT') +
                 MASK    (A  BWZ)
DT-LAST-SCHD-PAY 75   6  N                                  +
                 HEADING ('DATE LAST' 'SCHEDULED' 'PAYMENT') +
                 MASK    (A  BWZ)
DT-FIRST-PAY     81   6  N                             +
                 HEADING ('DATE FIRST' 'PAYMENT') +
                 MASK    (A  BWZ)
EXTENSIONS       87   2  N  0  HEADING ('LOAN' 'EXT') +
                 MASK    ('Z9')
RATE             89   4  N  2  HEADING ('LOAN' 'RATE')
REMAIN-MOS       93   3  N  0  HEADING ('REMAIN' 'MONTHS')
TERM             96   3  N  0  HEADING ('LOAN' 'TERM')
TOTAL-INT        99   4  P  2  HEADING ('TOTAL' 'INTEREST')
INT-EARNED      105   4  P  2  +
                 HEADING ('TOTAL' 'INTEREST' 'EARNED')
ADDRESS1        111  17  A     HEADING ('ADDRESS')
ADDRESS2        128  17  A     HEADING ('CITY / STATE')
ZIPCODE         145   5  N     HEADING ('ZIP CODE')
```

The data that the Case Study Three file generation routine (CS3FILGN) creates uses the previous file layout. The file ILMAST (installment loan master file) contains records 149 bytes long, with a block length of 14900 bytes. You must code JCL statements with the appropriate file characteristics when generating or accessing the ILMAST file. The macro CS3FILDF contains the previous file and field definitions.

Output

Tasks G and H create the output files ILATRIB and ILSTRAT, respectively. Task G performs an attributes sampling of ILMAST and writes the output file to ILATRIB. Task H performs a stratified random sampling of ILMAST and writes the output file to ILSTRAT.

Job Number One

You can combine tasks A through E into one execution of CA-PanAudit Plus, as follows:

```
%CS3FILDF
*
******  TASK A:
******  PERFORM AGING TO TEST FOR DELINQUENCY
*
%AGING1 ILMAST DT-LAST-SCHD-PAY MMDDYY LOAN-BAL
   IF STATCODE = 9 10
      GO TO JOB
   END-IF
%AGING2 ACCT-NBR LOAN-NBR
*
******  DEFINE SECOND JOB TO PERFORM TASKS B - F
*
JOB  INPUT ILMAST
*
******  BYPASS RECORDS WITH INVALID STATUS CODES
*
IF STAT-CODE = 9 10
```

```
   GO TO JOB
END-IF
*
******  TASK B:
******  SELECT RECORDS FOR EXTENSION REPORT
*
IF EXTENSIONS GT 5
   PRINT EXTENSION-REPORT
END-IF
*
******  TASK C:
******  SELECT INVALID RECORD TYPES
*
IF TYPE-CODE NE 1 THRU 8 10 12
   PRINT INVALID-TYPES
END-IF
*
******  TASK D:
******  CALCULATE ELAPSED DAYS FOR LAG TIME REPORT
*
DEFINE WS-LAG-DAYS    W    3    N     +
                      HEADING ('EXCESS' 'DAYS')
%DAYSCALC DT-FIRST-PAY MMDDYY LOAN-DATE MMDDYY WS-LAG-DAYS
IF WS-LAG-DAYS GT 45
   PRINT EXCESS-LAG-TIME
END-IF
*
******  TASK E:
******  CHECK IF PAST MATURITY DATE
*
%DAYSAGO MATUR-DATE MMDDYY GE 0
IF DAYSAGO-FLAG = 'YES'  AND  LOAN-BAL GT 0
     PRINT LATE-LOANS
END-IF
*
******  REPORTS INVOKED IN ABOVE CODING
*
REPORT  EXTENSION-REPORT  SPACE 2
   SEQUENCE   LOAN-OFCR ACCT-NBR LOAN-NBR
   CONTROL    LOAN-OFCR
   TITLE 01  'LOANS WITH MORE THAN 5 EXTENSIONS'
   TITLE 02  'AUDITED BANK OF BIG BANCSHARES'
   LINE  01   LOAN-OFCR      ACCT-NBR     LOAN-NBR    +
              NAME           LOAN-DATE    MATUR-DATE  +
              DT-LAST-PAY    ORIG-BAL     LOAN-BAL    +
              EXTENSIONS
*
REPORT  INVALID-TYPES  SPACE 2
  SEQUENCE   TYPE-CODE ACCT-NBR LOAN-NBR
  CONTROL    TYPE-CODE
  TITLE 01  'INVALID TYPE LOANS'
  TITLE 02  'AUDITED BANK OF BIG BANCSHARES'
  LINE  01   TYPE-CODE    ACCT-NBR     LOAN-NBR    +
             NAME         LOAN-DATE    MATUR-DATE  +
             ORIG-BAL     LOAN-BAL
*
REPORT  EXCESS-LAG-TIME  SPACE 2
  CONTROL   FINAL
  TITLE 01 'LOANS WITH MORE THAN 45 DAYS'
  TITLE 02 'BETWEEN ORIGINATION AND FIRST PAYMENT'
  TITLE 03 'AUDITED BANK OF BIG BANCSHARES'
  LINE  01  ACCT-NBR     LOAN-NBR        NAME      +
            LOAN-DATE    DT-FIRST-PAY    ORIG-BAL  +
            LOAN-BAL     WS-LAG-DAYS
*
REPORT  LATE-LOANS  SPACE 2
```

```
CONTROL   FINAL
TITLE 01 'LOANS WHICH HAVE REACHED MATURITY'
TITLE 02 'BUT HAVE NOT YET BEEN PAID OFF'
TITLE 03 'AUDITED BANK OF BIG BANCSHARES'
LINE  01  ACCT-NBR     LOAN-NBR      NAME          +
          LOAN-DATE    MATUR-DATE    DT-FIRST-PAY  +
          ORIG-BAL     LOAN-BAL      TERM
```

The first task performs the aging analysis while bypassing the appropriate status codes. The AGING routine defines the input file for its own internal use.

The next statement, JOB INPUT ILMAST, redefines the input file for the subsequent tasks. Then, statements are coded to bypass the appropriate status codes. This is followed by the code required to accomplish tasks B through E.

- Task B prints a line of the EXTENSION report if the number of extensions is greater than five.
- Task C prints a line of the INVALID-TYPE report if the TYPE-CODE field is nine or eleven.
- Task D first defines a work field (WS-LAG-DAYS). This field is used in the subsequent DAYSCALC routine to hold the number of days between the date of first payment and the loan date. The final three statements print a line of the EXCESS-LAG-TIME report if the field contains a value greater than 45.
- Task E uses the DAYSAGO routine to set the internal field, DAYSAGO-FLAG, to YES if the number of elapsed days between the maturity date and the current date is greater than or equal to zero. IF the DAYSAGO-FLAG contains the value YES and the LOAN-BAL field is greater than zero, a line of the LATE-LOANS report is printed.

The remainder of the code consists of the four reports mentioned previously. Each report defines the format of the listing by specifying items such as titles, sequencing of the report, and what fields are to be printed on each detail line.

Output - Task A

```
                                       CA-PANAUDIT PLUS AGING REPORT                                PAGE      1

                    FILE NAME: ILMAST    DATE FIELD: DT-LAST-SCHD-PAY    AMOUNT FIELD: LOAN-BAL

                                    RANGE 1       RANGE 2      RANGE 3

                                     31  -  60     61  -  90    OVER    90
                                    DAYS OLD      DAYS OLD     DAYS OLD

                                      AGING OF ACCOUNTS AS OF  3/01/90

          ACCOUNT     LOAN
 DATE      NUMBER    NUMBER         RANGE 1            RANGE 2            RANGE 3                TOTAL

1/29/90  00000437  00001014          12,857.14                .00                .00             12,857.14
         00000437                    12,857.14                .00                .00             12,857.14

1/29/90  00000462  00001039          34,800.00                .00                .00             34,800.00
         00000462                    34,800.00                .00                .00             34,800.00

10/12/89  00000493  00001070               .00                .00          54,158.33             54,158.33
```

	00000493		.00	.00	54,158.33	54,158.33
1/18/90	00000495	00001072	12,862.50	.00	.00	12,862.50
	00000495		12,862.50	.00	.00	12,862.50
1/29/90	00000504	00001081	40,659.26	.00	.00	40,659.26
	00000504		40,659.26	.00	.00	40,659.26
1/29/90	00000512	00001089	3,256.94	.00	.00	3,256.94
	00000512		3,256.94	.00	.00	3,256.94
1/29/90	00000524	00001101	1,140.00	.00	.00	1,140.00
	00000524		1,140.00	.00	.00	1,140.00
1/29/90	00000538	00001115	26,125.00	.00	.00	26,125.00
	00000538		26,125.00	.00	.00	26,125.00
10/03/89	00000541	00001118	.00	.00	22,611.11	22,611.11
	00000541		.00	.00	22,611.11	22,611.11
1/29/90	00000569	00001146	2,316.66	.00	.00	2,316.66
	00000569		2,316.66	.00	.00	2,316.66
12/02/89	00000577	00001154	.00	8,279.76	.00	8,279.76
	00000577		.00	8,279.76	.00	8,279.76
1/29/90	00000578	00001155	916.66	.00	.00	916.66
	00000578		916.66	.00	.00	916.66
10/11/89	00000592	00001169	.00	.00	4,433.33	4,433.33
	00000592		.00	.00	4,433.33	4,433.33
. .	.	.	.	.	.	.
. .	.	.	.	.	.	.
. .	.	.	.	.	.	.
1/29/90	00002576	00003153	10,483.33	.00	.00	10,483.33
	00002576		10,483.33	.00	.00	10,483.33
			1,226,307.27	429,073.59	819,670.15	2,475,051.01

NUMBER OF AGED RECORDS:	146
NUMBER OF CURRENT RECORDS:	1,824
NUMBER OF CURRENT AND AGED RECORDS:	1970
TOTAL AMOUNT OF CURRENT RECORDS:	30,951,175.50
TOTAL AMOUNT OF CURRENT AND AGED RECORDS:	33,426,226.51

The AGING report lists all loans which are delinquent as of the date the report is run. It lists the last scheduled payment date, the account and loan number, and the range which identifies the age of the delinquent loan. Final totals are included at the bottom of the report.

Output - Task B

LOANS WITH MORE THAN 5 EXTENSIONS PAGE 1
AUDITED BANK OF BIG BANCSHARES

LOAN OFCR	ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	MATURITY DATE	DATE LAST PAYMENT	ORIGINAL BALANCE	LOAN BALANCE	LOAN EXT
001	00000437	00001014	CUSTOMER15	8/01/87	7/31/94	1/20/90	20,000.00	12,857.14	7
	00000566	00001143	CUSTOMER144	6/28/85	6/28/93	2/16/90	12,800.00	5,333.33	9
	00001275	00001852	CUSTOMER853	7/16/86	7/15/89	2/01/90	30,800.00	5,988.88	7
	00002156	00002733	CUSTOMER1734	2/14/86	2/14/91	2/11/90	16,500.00	3,300.00	9
	00002288	00002865	CUSTOMER1866	4/19/88	4/19/90	2/16/90	43,100.00	3,591.66	9
	00002324	00002901	CUSTOMER1902	2/18/88	2/18/92	2/15/90	17,800.00	8,900.00	9

001							141,000.00	39,971.01	50
002	00000439	00001016	CUSTOMER17	5/12/89	5/11/96	2/08/90	7,700.00	6,875.00	6
	00000462	00001039	CUSTOMER40	1/30/90	1/30/91	1/17/90	34,800.00	34,800.00	8
	00000977	00001554	CUSTOMER555	6/14/87	6/13/92	2/08/90	12,900.00	6,020.00	7
	00001302	00001879	CUSTOMER880	12/02/89	12/02/94	1/20/90	47,400.00	45,820.00	7
	00001394	00001971	CUSTOMER972	3/17/87	3/17/91	2/10/90	41,700.00	11,293.75	6
	00001518	00002095	CUSTOMER1096	3/07/88	3/07/93	1/29/90	27,800.00	17,143.33	8
	00001637	00002214	CUSTOMER1215	11/18/86	11/17/92	2/13/90	52,700.00	24,154.16	7
	00001731	00002308	CUSTOMER1309	8/30/85	8/30/95	2/22/90	49,400.00	27,170.00	9
	00001785	00002362	CUSTOMER1363	5/28/89	5/28/98	2/12/90	18,300.00	16,775.00	8
	00001860	00002437	CUSTOMER1438	9/11/86	9/11/91	2/05/90	50,400.00	15,960.00	7
	00001879	00002456	CUSTOMER1457	6/28/87	6/27/97	2/17/90	46,200.00	33,880.00	6
	00001944	00002521	CUSTOMER1522	8/27/87	8/26/90	2/21/90	50,100.00	8,350.00	7
	00002218	00002795	CUSTOMER1796	12/06/88	12/06/96	2/03/90	19,900.00	16,997.91	7
	00002535	00003112	CUSTOMER2113	4/19/85	4/19/91	2/03/90	46,400.00	9,022.22	9
	00002602	00003179	CUSTOMER2180	7/07/87	7/06/96	1/31/90	32,400.00	23,100.00	9
002							538,100.00	297,361.37	111
003	00000975	00001552	CUSTOMER553	5/10/87	5/09/90	1/27/90	38,400.00	3,200.00	8
	00001507	00002084	CUSTOMER1085	1/09/88	1/08/94	2/01/90	57,700.00	37,665.28	7
	00001604	00002181	CUSTOMER1182	6/18/86	6/17/90	2/07/90	10,400.00	866.66	9
	00002091	00002668	CUSTOMER1669	11/12/87	11/11/94	2/04/90	36,900.00	25,039.28	7
	00002121	00002698	CUSTOMER1699	8/02/87	8/01/90	1/25/90	1,800.00	300.00	8
	00002150	00002727	CUSTOMER1728	5/31/87	5/30/93	2/24/90	51,200.00	27,733.33	7
003							196,400.00	94,804.55	46
004	00000491	00001068	CUSTOMER69	7/15/86	7/14/96	2/02/90	39,900.00	25,602.50	9
	00000845	00001422	CUSTOMER423	11/11/89	11/10/96	1/27/90	30,300.00	29,217.85	7
	00000990	00001567	CUSTOMER568	8/08/86	8/07/93	1/25/90	33,100.00	16,550.00	8
	00001273	00001850	CUSTOMER851	10/11/87	10/10/90	1/30/90	1,600.00	355.55	7
	00001285	00001862	CUSTOMER863	9/20/89	9/19/96	2/11/90	2,500.00	2,351.19	9
	00001478	00002055	CUSTOMER1056	7/24/85	7/23/89	2/11/90	28,000.00	4,083.33	7
	00001927	00002504	CUSTOMER1505	9/30/88	9/30/90	1/23/90	28,000.00	9,333.33	8
	00002021	00002598	CUSTOMER1599	7/18/86	7/17/89	2/11/90	39,000.00	7,583.33	7
004							202,400.00	95,077.08	62
005	00000459	00001036	CUSTOMER37	8/08/88	8/08/97	2/05/90	1,700.00	1,416.66	6
	00000819	00001396	CUSTOMER397	10/28/88	10/28/98	2/22/90	64,100.00	55,553.33	8
	00000987	00001564	CUSTOMER565	12/06/87	12/05/97	1/24/90	53,700.00	42,065.00	6
	00000988	00001565	CUSTOMER566	7/28/88	7/28/93	2/17/90	48,400.00	33,073.33	7
	00001164	00001741	CUSTOMER742	9/14/89	9/14/97	1/31/90	40,000.00	37,916.66	6
	00001178	00001755	CUSTOMER756	4/09/85	4/08/90	1/25/90	7,600.00	253.33	6
	00001209	00001786	CUSTOMER787	12/02/87	12/01/90	1/29/90	63,900.00	17,750.00	7
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
014	00000922	00001499	CUSTOMER500	8/05/86	8/04/90	1/28/90	60,900.00	7,612.50	7
	00001504	00002081	CUSTOMER1082	9/29/87	9/28/94	2/25/90	48,900.00	32,017.85	6
	00001724	00002301	CUSTOMER1302	10/08/85	10/08/94	2/03/90	34,000.00	17,629.63	6
	00001938	00002515	CUSTOMER1516	5/15/85	5/15/95	2/01/90	34,100.00	17,902.50	9
	00002000	00002577	CUSTOMER1578	5/07/88	5/07/96	2/03/90	26,400.00	20,625.00	7
	00002078	00002655	CUSTOMER1656	8/09/87	8/09/91	2/02/90	63,300.00	23,737.50	7
014							267,600.00	119,524.98	42
							3726,000.00	1769,256.03	787

The EXTENSION report lists the appropriate fields for each loan which has been extended more than five times. The report is summarized by loan officer and appropriate totals are listed.

Output - Task C

INVALID TYPE LOANS PAGE 1
AUDITED BANK OF BIG BANCSHARES

ACCT TYPE CODE	ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	MATURITY DATE	ORIGINAL BALANCE	LOAN BALANCE
09	00000466	00001043	CUSTOMER44	6/22/87	6/21/92	27,700.00	12,926.66
	00000520	00001097	CUSTOMER98	12/20/85	12/19/90	49,000.00	8,166.67
	00000592	00001169	CUSTOMER170	6/12/88	6/12/90	13,300.00	4,433.33
	00000610	00001187	CUSTOMER188	5/15/86	5/15/94	27,900.00	15,984.37
	00000628	00001205	CUSTOMER206	9/10/89	9/10/94	45,800.00	44,273.33
	00000952	00001529	CUSTOMER530	4/28/88	4/28/91	19,400.00	7,544.44
	00000988	00001565	CUSTOMER566	7/28/88	7/28/93	48,400.00	33,073.33
	00001042	00001619	CUSTOMER620	9/19/89	9/19/90	36,700.00	21,408.33
	00001294	00001871	CUSTOMER872	10/03/85	10/03/93	43,700.00	20,029.16
	00001312	00001889	CUSTOMER890	8/13/85	8/12/90	20,100.00	2,010.00
	00001474	00002051	CUSTOMER1052	5/23/85	5/22/90	38,600.00	1,930.00
	00001546	00002123	CUSTOMER1124	8/17/85	8/17/90	2,500.00	250.00
	00001564	00002141	CUSTOMER1142	5/18/86	5/17/92	24,900.00	9,337.50
	00001654	00002231	CUSTOMER1232	1/28/89	1/28/99	56,700.00	50,557.50
	00001672	00002249	CUSTOMER1250	5/23/89	5/23/90	23,800.00	5,950.00
	00001798	00002375	CUSTOMER1376	10/14/88	10/14/93	7,300.00	5,353.33
	00001888	00002465	CUSTOMER1466	12/17/85	12/16/90	26,000.00	4,333.33
	00001924	00002501	CUSTOMER1502	3/31/89	3/31/97	8,100.00	7,340.62
	00001960	00002537	CUSTOMER1538	10/25/88	10/25/92	64,800.00	43,200.00
	00001996	00002573	CUSTOMER1574	9/16/85	9/16/90	40,900.00	4,771.67
	00002086	00002663	CUSTOMER1664	12/12/88	12/12/95	34,300.00	28,583.33
	00002140	00002717	CUSTOMER1718	1/28/86	1/28/94	33,800.00	17,252.08
	00002158	00002735	CUSTOMER1736	11/14/85	11/14/95	26,600.00	15,295.00
	00002194	00002771	CUSTOMER1772	5/27/87	5/26/93	37,300.00	20,204.16
	00002212	00002789	CUSTOMER1790	7/10/85	7/10/94	27,700.00	13,593.52
	00002230	00002807	CUSTOMER1808	1/12/88	1/12/96	59,500.00	45,864.58
	00002248	00002825	CUSTOMER1826	8/04/86	8/03/90	37,200.00	4,650.00
	00002392	00002969	CUSTOMER1970	9/08/88	9/08/90	37,100.00	10,820.83
	00002410	00002987	CUSTOMER1988	12/18/85	12/17/90	31,700.00	5,283.33
	00002500	00003077	CUSTOMER2078	1/21/87	1/20/97	19,300.00	13,349.16
	00002554	00003131	CUSTOMER2132	7/18/88	7/18/90	11,600.00	2,416.66
	00002608	00003185	CUSTOMER2186	6/24/85	6/24/94	15,600.00	7,511.11
09						997,300.00	487,697.33
11	00000718	00001295	CUSTOMER296	12/28/87	12/27/89	61,100.00	5,091.66
	00000844	00001421	CUSTOMER422	3/08/87	3/07/96	57,700.00	39,000.92
	00000862	00001439	CUSTOMER440	12/25/89	12/25/90	62,100.00	51,750.00
	00000934	00001511	CUSTOMER512	8/13/88	8/13/90	17,500.00	4,375.00
	00001078	00001655	CUSTOMER656	1/26/90	1/25/97	26,100.00	25,789.28
	00001186	00001763	CUSTOMER764	10/07/86	10/07/91	43,100.00	14,366.66
	00001222	00001799	CUSTOMER800	12/21/85	12/21/90	59,300.00	9,883.33
	00001330	00001907	CUSTOMER908	11/10/89	11/10/97	6,900.00	6,684.37
	00001402	00001979	CUSTOMER980	6/11/88	6/11/93	61,100.00	40,733.33
	00001510	00002087	CUSTOMER1088	7/10/85	7/09/92	15,600.00	5,385.71
	00001636	00002213	CUSTOMER1214	2/01/89	2/01/95	1,200.00	1,000.00
	00002050	00002627	CUSTOMER1628	6/07/87	6/06/90	31,200.00	3,466.66
	00002068	00002645	CUSTOMER1646	11/03/88	11/03/97	23,300.00	20,063.88
	00002302	00002879	CUSTOMER1880	2/07/88	2/07/92	25,800.00	12,900.00
	00002320	00002897	CUSTOMER1898	10/25/86	10/25/91	56,300.00	18,766.66

INVALID TYPE LOANS PAGE 2
AUDITED BANK OF BIG BANCSHARES

ACCT TYPE CODE	ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	MATURITY DATE	ORIGINAL BALANCE	LOAN BALANCE

11		548,300.00	259,257.46
		1545,600.00	746,954.79

The INVALID TYPE LOANS report lists the appropriate fields for type codes of nine and eleven. It lists all invalid type loans and contains totals for the original balance and the loan balance.

Output - Task D

LOANS WITH MORE THAN 45 DAYS PAGE 1
BETWEEN ORIGINATION AND FIRST PAYMENT
AUDITED BANK OF BIG BANCSHARES

ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	DATE FIRST PAYMENT	ORIGINAL BALANCE	LOAN BALANCE	EXCESS DAYS
00000502	00001079	CUSTOMER80	1/10/88	3/20/88	60,400.00	39,427.78	070
00000538	00001115	CUSTOMER116	12/01/87	2/20/88	57,000.00	26,125.00	081
00000741	00001318	CUSTOMER319	2/03/86	4/22/86	36,100.00	18,050.00	078
00000742	00001319	CUSTOMER320	8/25/88	10/16/88	6,000.00	4,500.00	052
00000796	00001373	CUSTOMER374	6/06/88	8/21/88	42,600.00	30,766.66	076
00000871	00001448	CUSTOMER449	11/17/89	1/08/90	43,900.00	41,705.00	052
00000882	00001459	CUSTOMER460	4/03/88	6/26/88	3,500.00	2,430.55	084
00001050	00001627	CUSTOMER628	10/22/89	1/04/90	40,200.00	37,966.66	074
00001066	00001643	CUSTOMER644	7/11/85	10/01/85	60,200.00	5,016.66	082
00001198	00001775	CUSTOMER776	11/25/85	1/31/86	48,700.00	28,002.50	067
00001212	00001789	CUSTOMER790	10/08/87	12/12/87	27,000.00	18,000.00	065
00001278	00001855	CUSTOMER856	11/10/87	12/26/87	12,900.00	9,271.87	046
00001293	00001870	CUSTOMER871	8/31/86	11/27/86	23,000.00	7,283.33	088
00001439	00002016	CUSTOMER1017	11/05/89	1/07/90	22,500.00	20,625.00	063
00001464	00002041	CUSTOMER1042	5/05/87	7/11/87	46,900.00	21,105.00	067
00001614	00002191	CUSTOMER1192	3/14/87	5/05/87	52,300.00	21,791.66	052
00001617	00002194	CUSTOMER1195	3/08/89	5/12/89	23,400.00	12,675.00	065
00001678	00002255	CUSTOMER1256	2/17/86	5/11/86	61,900.00	20,633.33	083
00001826	00002403	CUSTOMER1404	10/07/87	11/30/87	2,900.00	1,208.33	054
00001830	00002407	CUSTOMER1408	12/10/88	2/10/89	25,900.00	22,122.91	062
00001953	00002530	CUSTOMER1531	1/12/90	4/01/90	16,800.00	16,566.66	079
00001978	00002555	CUSTOMER1556	6/29/89	8/15/89	64,900.00	54,083.33	047
00002163	00002740	CUSTOMER1741	10/02/89	12/12/89	36,500.00	30,416.66	071
00002169	00002746	CUSTOMER1747	8/02/87	9/29/87	55,000.00	9,166.66	058
00002176	00002753	CUSTOMER1754	4/15/86	6/01/86	41,700.00	21,718.75	047
00002347	00002924	CUSTOMER1925	7/09/89	9/30/89	37,000.00	31,604.16	083
00002414	00002991	CUSTOMER1992	11/28/89	2/21/90	21,900.00	19,162.50	085
00002489	00003066	CUSTOMER2067	7/12/85	9/20/85	52,900.00	4,408.33	070
00002611	00003188	CUSTOMER2189	2/26/90	4/20/90	58,000.00	58,000.00	053
					1082,000.00	633,834.29	

The EXCESS-LAG-TIME report lists all loans with more than 45 days between the date of first payment and the loan date. Appropriate fields are listed, including the number of days before the first payment.

Output - Task E

LOANS WHICH HAVE REACHED MATURITY PAGE 1
BUT HAVE NOT YET BEEN PAID OFF
AUDITED BANK OF BIG BANCSHARES

ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	MATURITY DATE	DATE FIRST PAYMENT	ORIGINAL BALANCE	LOAN BALANCE	LOAN TERM
00000540	00001117	CUSTOMER118	12/16/85	12/15/89	12/31/85	38,600.00	1,608.33	48
00000718	00001295	CUSTOMER296	12/28/87	12/27/89	1/01/88	61,100.00	5,091.66	24

00000815	00001392	CUSTOMER393	11/18/86	11/17/89	11/19/86	26,200.00	2,183.33	36
00001232	00001809	CUSTOMER810	10/10/86	10/09/89	10/28/86	41,000.00	4,555.55	36
00001266	00001843	CUSTOMER844	1/21/88	1/20/90	2/04/88	59,300.00	2,470.83	24
00001275	00001852	CUSTOMER853	7/16/86	7/15/89	8/09/86	30,800.00	5,988.88	36
00001303	00001880	CUSTOMER881	11/19/85	11/18/89	11/28/85	59,100.00	3,693.75	48
00001374	00001951	CUSTOMER952	1/28/87	1/27/90	1/30/87	2,200.00	61.11	36
00001478	00002055	CUSTOMER1056	7/24/85	7/23/89	7/28/85	28,000.00	4,083.33	48
00001513	00002090	CUSTOMER1091	1/06/86	1/05/90	1/28/86	55,900.00	1,164.58	48
00001558	00002135	CUSTOMER1136	8/23/85	8/22/89	8/26/85	19,400.00	2,424.99	48
00001911	00002488	CUSTOMER1489	7/28/86	7/27/89	8/13/86	58,700.00	11,413.88	36
00001926	00002503	CUSTOMER1504	12/06/85	12/05/89	12/14/85	11,400.00	475.00	48
00001988	00002565	CUSTOMER1566	10/02/85	10/01/89	10/02/85	30,700.00	2,558.33	48
00002021	00002598	CUSTOMER1599	7/18/86	7/17/89	8/02/86	39,000.00	7,583.33	36
00002053	00002630	CUSTOMER1631	8/16/86	8/15/89	9/04/86	60,800.00	10,133.32	36
00002060	00002637	CUSTOMER1638	9/17/86	9/16/89	9/19/86	34,800.00	4,833.33	36
00002147	00002724	CUSTOMER1725	8/29/85	8/28/89	9/18/85	37,500.00	4,687.50	48
00002234	00002811	CUSTOMER1812	3/01/89	3/01/90	3/09/89	23,600.00	1,966.66	12
00002393	00002970	CUSTOMER1971	1/05/88	1/04/90	1/13/88	63,900.00	2,662.50	24
00002526	00003103	CUSTOMER2104	8/08/86	8/07/89	8/31/86	3,300.00	549.99	36
00002605	00003182	CUSTOMER2183	10/28/86	10/27/89	11/06/86	56,500.00	6,277.77	36
						841,800.00	86,467.95	828

The LATE-LOANS report lists the appropriate fields for all loans which have reached the maturity date and have not been paid off.

Now that audit tasks A through E have been accomplished, the resident auditor asks the EDP auditor for help on tasks F through G:

F. Recompute earned/unearned interest to ensure that the right amount was taken into earnings.

G. Take an attribute sample and examine loan files to ensure that all of the proper documents (note, collateral assignment, and so on) are present.

H. Take a stratified monetary sample and send out positive confirmations.

The EDP auditor evaluates the scope of each task and decides that they will all exist as separate CA-PanAudit Plus jobs.

Job Number Two

To accomplish task F, the EDP auditor contacts the loan department and is told that interest is legally earned according to the Rule of 78s. He also finds that, according to a book containing a collection of financial formulas, the earnings formula for the Rule of 78s is:

$$IE = T - T \frac{r(r - 1)}{n(n + 1)}$$

Therefore, the EDP auditor constructs the following job to check if the earned interest on file is within $.50 of the amount calculated by the formula for the Rule of 78s. This job does not contain an invocation for any CA-PanAudit Plus routine and consists entirely of code from the host language, CA-Easytrieve Plus .

```
%CS3FILDF
*
******  TASK F:
******  CALCULATE EARNED INTEREST AND COMPARE FILE TOTAL
*
JOB INPUT ILMAST
IF STAT-CODE EQ 9 10
   GO TO JOB
END-IF
DEFINE CALC-INT-ERND       W     4    P 2           +
             HEADING  ('RECALCULATED' 'INTEREST' 'EARNED')
DEFINE REBATE-FACT         W     6    P 4
DEFINE DIFFERENCE          W     4    P 2
  REBATE-FACT   = (REMAIN-MOS * (REMAIN-MOS  -1))      +
                  / (TERM * (TERM + 1))
  CALC-INT-ERND = TOTAL-INT  (TOTAL-INT * REBATE-FACT)
  DIFFERENCE    = INT-EARNED – CALC-INT-ERND
  IF DIFFERENCE GT 0.5 OR DIFFERENCE LT -0.5
     PRINT INT-DIFFERENCE
  END-IF
*
******  EARNED INTEREST REPORT
*
REPORT  INT-DIFFERENCE  SPACE 0
  SEQUENCE  DIFFERENCE ACCT-NBR LOAN-NBR
  CONTROL   FINAL
  TITLE 01 'LOANS WITH EARNED INTEREST NOT MATCHING'
  TITLE 02 'RECALCULATED EARNED INTEREST'
  TITLE 03 'AUDITED BANK OF BIG BANCSHARES'
  LINE  01  ACCT-NBR   +1   LOAN-NBR     +1   +
            NAME            ORIG-BAL     +
            LOAN-BAL        RATE         +
            TERM            REMAIN-MOS   +
            TOTAL-INT       INT-EARNED   +
            CALC-INT-ERND   DIFFERENCE
```

This job begins with the customary definition of the input file and bypassing of records. This is followed by definition of work fields for the various calculations. Then, the calculated interest earned is subtracted from the interest earned field recorded in the installment loan master file. If the numbers differ more than $.50, a line of the INT-DIFFERENCE report is generated.

Output - Task F

LOANS WITH EARNED INTEREST NOT MATCHING PAGE 1

RECALCULATED EARNED INTEREST

AUDITED BANK OF BIG BANCSHARES

ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	ORIGINAL BALANCE	LOAN BALANCE	LOAN RATE	LOAN TERM	REMAIN MONTHS	TOTAL INTEREST	TOTAL INTEREST EARNED	RECALCULATED INTEREST EARNED	DIFFERENCE
00001421	00001998	CUSTOMER999	4,700.00	3,981.94	13.00	72	61	1,858.21	571.33	564.33	7.00
00000539	00001116	CUSTOMER117	22,700.00	13,241.66	8.00	60	35	4,615.56	3,123.04	3,115.04	8.00
00000490	00001067	CUSTOMER68	54,600.00	1,137.50	8.00	48	1	8,917.99	8,936.99	8,917.99	19.00
00000613	00001190	CUSTOMER191	4,100.00	4,024.07	17.00	108	106	3,165.35	195.82	172.82	23.00
00000473	00001050	CUSTOMER51	35,100.00	4,095.00	12.00	60	7	10,705.50	10,613.45	10,583.45	30.00
00000596	00001173	CUSTOMER174	33,600.00	4,200.00	16.00	48	6	10,975.99	10,870.59	10,836.59	34.00
00002001	00002578	CUSTOMER1579	10,000.00	6,354.16	9.00	96	61	3,637.49	2,249.95	2,207.95	42.00
00001791	00002368	CUSTOMER1369	35,600.00	4,450.00	12.00	48	6	8,721.99	8,668.22	8,611.22	57.00
00002057	00002634	CUSTOMER1635	47,100.00	28,129.16	14.00	72	43	20,056.74	13,223.24	13,165.24	58.00
00002585	00003162	CUSTOMER2163	55,300.00	42,396.66	15.00	120	92	41,820.62	17,773.03	17,711.03	62.00
00002581	00003158	CUSTOMER2159	24,000.00	13,400.00	11.00	120	67	13,309.99	9,324.09	9,257.09	67.00
00001318	00001895	CUSTOMER896	61,600.00	31,940.74	17.00	108	56	47,559.96	35,186.27	35,118.27	68.00
00001557	00002134	CUSTOMER1135	6,100.00	5,809.52	10.00	84	80	2,160.27	319.21	248.21	71.00

00000581	00001158	CUSTOMER159	62,000.00	54,250.00	16.00	48	42	20,253.16	5,507.82	5,425.82	82.00
00002311	00002888	CUSTOMER1889	1,900.00	475.00	15.00	48	12	581.87	632.22	549.22	83.00
00002194	00002771	CUSTOMER1772	37,300.00	20,204.16	15.00	72	39	17,018.12	12,306.71	12,220.71	86.00
00000520	00001097	CUSTOMER98	49,000.00	8,166.67	10.00	60	10	12,454.06	12,235.93	12,148.93	87.00
00002195	00002772	CUSTOMER1773	39,200.00	21,777.77	15.00	36	20	9,064.99	6,571.65	6,479.65	92.00
00000710	00001287	CUSTOMER288	53,900.00	23,741.66	13.00	84	37	24,816.17	20,292.95	20,187.95	105.00
00002003	00002580	CUSTOMER1581	25,600.00	12,444.44	15.00	72	35	11,679.99	9,144.64	9,035.64	109.00
00000616	00001193	CUSTOMER194	22,600.00	18,080.00	15.00	60	48	8,616.24	3,422.05	3,306.05	116.00
00001992	00002569	CUSTOMER1570	38,100.00	19,685.00	11.00	60	31	10,652.12	8,071.48	7,946.48	125.00
00002186	00002763	CUSTOMER1764	13,500.00	10,875.00	14.00	36	29	2,913.75	1,267.52	1,137.52	130.00
00000432	00001009	CUSTOMER10	8,600.00	4,300.00	16.00	60	30	3,497.13	2,797.86	2,665.86	132.00
00001319	00001896	CUSTOMER897	54,500.00	4,541.66	17.00	48	4	18,915.96	18,954.48	18,819.48	135.00
00001443	00002020	CUSTOMER1021	56,100.00	49,866.66	11.00	72	64	18,770.11	4,526.55	4,371.55	155.00
00001288	00001865	CUSTOMER866	54,500.00	26,493.05	8.00	72	35	13,261.54	10,423.12	10,259.12	164.00
00000671	00001248	CUSTOMER249	32,300.00	14,227.38	14.00	84	37	16,015.27	13,196.42	13,028.42	168.00
00000509	00001086	CUSTOMER87	23,800.00	7,272.22	10.00	36	11	3,669.10	3,546.39	3,366.39	180.00
00000767	00001344	CUSTOMER345	25,400.00	21,469.04	16.00	84	71	14,393.04	4,563.48	4,375.48	188.00
00001509	00002086	CUSTOMER1087	45,500.00	5,055.55	8.00	36	4	5,611.60	5,752.09	5,561.09	191.00
			1038,300.00	486,085.67	401.00	2112	1180	389,689.88	264,268.59	261,394.59	2,874.00

The final three columns of this report list the total interest earned recorded in the input file, the recalculated interest earned computed in the job, and the difference between these figures.

Job Number Three

You can accomplish task G by taking an attribute sample of the file and printing a report listing the sample records. The code that you require to perform this task is:

```
%CS3FILDF
*
******  DEFINE OUTPUT FILE FOR ATTRIBUTE SAMPLE
*
FILE ILATRIB F (149)
   COPY ILMAST
*
******  OBTAIN EXACT POPULATION COUNT FOR ATTRIBUTES SAMPLING
*
JOB INPUT ILATRIB
DEFINE NUM-RECS    S   4   P  0    VALUE (0)
IF STAT-CODE = 9 10
   GO TO JOB
ELSE
   NUM-RECS = NUM-RECS + 1
END-IF
*
******  TASK G:
******  PERFORM ATTRIBUTE SAMPLING
*
%ATTSAMP1 ILMAST NUM-RECS 95 3 2 259871
   IF STAT-CODE = 9 10
      GO TO JOB
   END-IF
%ATTSAMP2 ILATRIB
*
******  PRINT ATTRIBUTE SAMPLE REPORT
*
JOB  INPUT ILATRIB
   PRINT ATT-SAMPLE
*
******  ATTRIBUTE SAMPLE REPORT
```

```
*
REPORT  ATT-SAMPLE  SPACE 2
  CONTROL   FINAL
  TITLE 01 'ATTRIBUTE SAMPLE FOR VERIFICATION OF'
  TITLE 02 'REQUIRED LOAN DOCUMENTATION'
  TITLE 03 'AUDITED BANK OF BIG BANCSHARES'
  LINE  01  ACCT-NBR      LOAN-NBR      +
            NAME          LOAN-DATE     +
            MATUR-DATE    ORIG-BAL      +
            LOAN-BAL
```

The routine contains a FILE statement for the sample file that ATTSAMP creates. The COPY ILMAST statement specifies that the field names, lengths, and attributes generated for the ILATRIB file are identical to those in the ILMAST file.

The next section of code is an example of logic before the invocation of a stand-alone routine. The subsequent ATTSAMP routine requires an exact population count as a parameter. To obtain an exact count, an initial job is coded to accumulate the exact count in the NUM-RECS field. This job bypasses records with a status code of 9 or 10 and increments a counter for all other records. The value calculated for NUM-RECS is specified as the size parameter in the subsequent ATTSAMP routine. This allows ATTSAMP to execute with a dynamically calculated size parameter and ensures that the correct number of records are written to the sample file.

The ATTSAMP routine specifies a 95 percent confidence, 3 percent precision, 2 percent error rate, and the dynamically calculated value for population size. ATTSAMP then writes the appropriate number of records to the ILATRIB file.

The ILATRIB file is input to the next job, which prints a report listing the appropriate fields for all records in the sample file selected by ATTSAMP.

Output Task - G

The following is the report generated from the first job which selected the sample from the installment loan master file.

```
                 ATTRIBUTE SAMPLING REPORT
                     INPUT PARAMETERS

        INPUT FILENAME                            ILMAST
        TOTAL POPULATION SIZE                      1,970
        REQUIRED PRECISION                             3.00
        REQUIRED CONFIDENCE LEVEL                     95
        ERROR RATE                                     2.00

                     SAMPLE RESULTS

        SAMPLE PERCENTAGE REQUIRED                     4.06091370%
        SAMPLE SIZE REQUIRED                          80
```

```
                  SAMPLE FILE

NUMBER OF RECORDS PROCESSED              1,970
NUMBER OF RECORDS REQUESTED                 80
NUMBER OF RECORDS IN SAMPLE FILE            80

FILE ILATRIB  WILL BE CREATED
```

This report lists the input parameters and the sample results. From the values input to ATTSAMP, it is determined that 4.06091370 percent of the file constituted a representative sample which dictates a sample file size of 80 records. The final section of the report indicates that the correct number of records were written to the ILATRIB file.

The next report was generated from job number three's second job and lists appropriate fields from the records in the attribute sample file.

ATTRIBUTE SAMPLE FOR VERIFICATION OF   PAGE 1
REQUIRED LOAN DOCUMENTATION
AUDITED BANK OF BIG BANCSHARES

ACCOUNT NUMBER	LOAN NUMBER	BORROWER NAME	LOAN ISSUE DATE	MATURITY DATE	ORIGINAL BALANCE	LOAN BALANCE
00000430	00001007	CUSTOMER8	11/06/85	11/06/95	23,500.00	13,512.50
00000478	00001055	CUSTOMER56	3/10/88	3/10/98	1,500.00	1,212.50
00000487	00001064	CUSTOMER65	7/14/86	7/13/92	21,400.00	8,619.44
00000503	00001080	CUSTOMER81	9/05/88	9/05/92	16,200.00	10,462.50
00000624	00001201	CUSTOMER202	11/20/89	11/19/96	51,500.00	49,660.71
00000627	00001204	CUSTOMER205	11/16/89	11/16/91	43,100.00	37,712.50
00000675	00001252	CUSTOMER253	9/09/89	9/09/90	3,600.00	2,100.00
00000767	00001344	CUSTOMER345	1/29/89	1/29/96	25,400.00	21,469.04
00000769	00001346	CUSTOMER347	1/10/89	1/10/95	27,700.00	22,698.61
00000781	00001358	CUSTOMER359	8/12/86	8/11/90	60,700.00	7,587.50
00000803	00001380	CUSTOMER381	1/17/88	1/16/91	61,500.00	18,791.66
00000919	00001496	CUSTOMER497	9/24/88	9/24/90	11,100.00	3,237.50
00000937	00001514	CUSTOMER515	3/25/85	3/24/90	16,300.00	271.67
00000993	00001570	CUSTOMER571	5/19/89	5/19/97	54,300.00	49,209.37
00001009	00001586	CUSTOMER587	10/05/89	10/05/90	34,600.00	23,066.66
00001025	00001602	CUSTOMER603	9/14/86	9/13/90	25,300.00	3,689.58
00001028	00001605	CUSTOMER606	6/16/89	6/16/95	44,600.00	39,644.44
00001054	00001631	CUSTOMER632	6/24/87	6/23/97	35,300.00	25,886.66
00001070	00001647	CUSTOMER648	8/12/89	8/12/90	53,300.00	31,091.66
00001101	00001678	CUSTOMER679	8/01/87	7/31/90	2,800.00	466.66
00001121	00001698	CUSTOMER699	1/25/89	1/25/95	3,800.00	3,113.88
00001150	00001727	CUSTOMER728	4/02/86	4/01/90	16,300.00	679.16
00001161	00001738	CUSTOMER739	2/07/88	2/07/96	36,900.00	27,675.00
00001174	00001751	CUSTOMER752	8/12/86	8/11/93	48,200.00	24,100.00
00001183	00001760	CUSTOMER761	10/06/85	10/06/90	42,000.00	5,600.00
00001193	00001770	CUSTOMER771	12/03/88	12/03/96	13,000.00	11,104.16
00001271	00001848	CUSTOMER849	11/05/88	11/05/91	56,100.00	32,725.00
00001275	00001852	CUSTOMER853	7/16/86	7/15/89	30,800.00	5,988.88
00001294	00001871	CUSTOMER872	10/03/85	10/03/93	43,700.00	20,029.16
00001301	00001878	CUSTOMER879	3/16/88	3/16/90	4,400.00	183.33
00001351	00001928	CUSTOMER929	2/19/90	1/27/91	33,200.00	33,200.00
00001394	00001971	CUSTOMER972	3/17/87	3/17/91	41,700.00	11,293.75
00001417	00001994	CUSTOMER995	3/14/85	3/14/90	25,600.00	426.67
00001482	00002059	CUSTOMER1060	5/08/89	5/08/98	60,600.00	56,111.11
00001493	00002070	CUSTOMER1071	4/06/89	4/06/93	53,300.00	42,195.83
00001536	00002113	CUSTOMER1114	12/31/88	12/31/98	46,700.00	41,640.83
00001566	00002143	CUSTOMER1144	4/27/85	4/26/90	5,600.00	186.66
00001594	00002171	CUSTOMER1172	9/10/88	9/10/94	60,700.00	46,368.05

```
00001669  00002246  CUSTOMER1247    12/29/88   12/29/93     48,700.00      37,336.66
00001678  00002255  CUSTOMER1256     2/17/86    2/17/92     61,900.00      20,633.33
00001696  00002273  CUSTOMER1274     8/03/85    8/03/94     38,200.00      19,100.00
00001738  00002315  CUSTOMER1316     6/18/87    6/17/92      9,400.00       4,386.66
00001800  00002377  CUSTOMER1378    12/22/89   12/21/92     12,700.00      11,994.44
00001836  00002413  CUSTOMER1414     7/06/86    7/05/90     11,000.00       1,145.83
00001867  00002444  CUSTOMER1445     4/22/85    4/22/94     17,100.00       7,916.66
00001869  00002446  CUSTOMER1447    10/17/89   10/17/98     11,900.00      11,459.25
00001877  00002454  CUSTOMER1455     1/21/88    1/20/91     55,300.00      16,897.22
00001882  00002459  CUSTOMER1460     7/04/85    7/03/90     62,800.00       5,233.33
00001888  00002465  CUSTOMER1466    12/17/85   12/16/90     26,000.00       4,333.33
    .        .        .         .        .          .           .
    .        .        .         .        .          .           .
    .        .        .         .        .          .           .
00002367  00002944  CUSTOMER1945    12/12/89   12/11/96     53,800.00      52,519.04
00002433  00003010  CUSTOMER2011     2/06/86    2/05/91     23,500.00       4,700.00
00002435  00003012  CUSTOMER2013     1/12/90    1/12/95     31,300.00      30,778.33
00002437  00003014  CUSTOMER2015     1/03/90    1/03/91     38,700.00      35,475.00
00002444  00003021  CUSTOMER2022     6/17/86    6/16/90     33,100.00       2,758.33
00002452  00003029  CUSTOMER2030    10/31/85   10/31/91     17,800.00       5,191.66
00002475  00003052  CUSTOMER2053     6/07/87    6/06/90      5,000.00         555.55
00002549  00003126  CUSTOMER2127     4/03/88    4/03/90     53,000.00       4,416.66
00002551  00003128  CUSTOMER2129    12/06/86   12/05/96     34,500.00      23,575.00
00002587  00003164  CUSTOMER2165     7/09/88    7/09/97     49,500.00      40,791.66
                                                          2440,700.00    1342,238.73
```

Job Number Four

Task H performs a stratified monetary sampling and prints a report of all records in the sample, and a positive confirmation letter for each item in the sample. The coding to accomplish this is as follows:

```
%CS3FILDF
*
******  DEFINE OUTPUT FILE FOR STRATIFIED SAMPLE
*
FILE ILSTRAT F(149)
   COPY ILMAST
*
******  TASK H:
******  PERFORM STRATIFIED SAMPLING
*
%STRATIF1 ILMAST LOAN-BAL 5000000 95 1250000 50000 15
   IF STAT-CODE = 9 10
      GO TO STRAT-JOB
   END-IF
%STRATIF2 ILSTRAT
*
******  PRINT STRATIFIED SAMPLING REPORTS
*
JOB  INPUT ILSTRAT
   PRINT CONFIRMATION-CONTROL
   PRINT CONFIRMATIONS
*
******  CONFIRMATION CONTROL REPORT
*
REPORT  CONFIRMATION-CONTROL  SKIP 1
  CONTROL   FINAL
  TITLE 01 'LOANS SELECTED FOR POSITIVE CONFIRMATION'
  TITLE 02 'USING STRATIFIED MONETARY SAMPLING METHOD'
  TITLE 03 'AUDITED BANK OF BIG BANCSHARES'
  LINE  01  ACCT-NBR     LOAN-NBR      NAME    +10   +
            LOAN-DATE    MATUR-DATE    ORIG-BAL      +
```

```
            LOAN-BAL
  LINE  02  POS 3 ADDRESS1
  LINE  03  POS 3 ADDRESS2 -2 ZIPCODE
*
******  CONFIRMATION LETTERS
*
REPORT  CONFIRMATIONS NOADJUST NODATE NOPAGE LINESIZE 70
  SEQUENCE  ACCT-NBR
  CONTROL   FINAL NEWPAGE ACCT-NBR NEWPAGE NOPRINT
TITLE 01  COL 20 'AUDITED BANK OF BIG BANCSHARES'
TITLE 02  COL 25 '1269 SUNRAY PARKWAY'
TITLE 03  COL 22 'SUNSHINE, FLORIDA  35060'
TITLE 06  COL 50 'JULY 28,1985'
TITLE 10  COL 5 NAME
TITLE 11  COL 5 ADDRESS1
TITLE 12  COL 5 ADDRESS2 -2 ZIPCODE
TITLE 14  COL 5 'GENTLEMEN:'
TITLE 16  COL 5 'WE ARE IN THE PROCESS OF AUDITING '  +
                'INSTALLMENT LOANS'
TITLE 17  COL 5 'AND WOULD LIKE FOR YOU TO VERIFY '  +
                'INFORMATION ABOUT'
TITLE 18  COL 5 'YOUR ACCOUNT. AS OF JULY 28, 1985 '  +
                'OUR RECORDS'
TITLE 19  COL 5 'SHOW THE INFORMATION LISTED BELOW. '  +
                'NOTE THAT THIS'
TITLE 20  COL 5 'CONFIRMATION IS ONLY FOR THE LOAN '  +
                'INDICATED, EVEN'
TITLE 21  COL 5 'THOUGH YOU MAY HAVE OTHER LOANS.'
TITLE 23  COL 5 'PLEASE INDICATE ON THE REVERSE SIDE '  +
                'OF THIS LETTER'
TITLE 24  COL 5 'WHETHER OR NOT YOU AGREE WITH THE '  +
                'INFORMATION LISTED'
TITLE 25  COL 5 'BELOW. IF YOUR RECORDS ARE '  +
                'DIFFERENT, PLEASE'
TITLE 26  COL 5 'INDICATE. RETURN THIS LETTER IN '  +
                'THE ENCLOSED,'
TITLE 27  COL 5 'POSTAGE PAID ENVELOPE.'
LINE  01  COL 10 ACCT-NBR LOAN-NBR LOAN-BAL MATUR-DATE
LINE  04  COL 5 'THANK YOU FOR YOUR COOPERATION.'
LINE  09  COL 30 'SINCERELY,'
LINE  10  COL 30 'AUDITED BANK OF BIG BANCSHARES'
```

The program begins with the customary input file and definition of the sample file to be written by STRATIF. The STRATIF1 routine is coded with 5,000,000 as the stratum size, a confidence of 95 percent, precision of 1,250,000, and materiality of 50,000. As usual, the invalid status codes of nine and ten are bypassed from STRATIF processing.

Output - Task H

The following report is generated from the execution of the STRATIF routine:

```
                    RESULTS OF STRATIFIED SAMPLING                        PAGE    1
             INPUT FILENAME: ILMAST    INPUT FIELD: LOAN-BAL
             STRATUM SIZE:         5,000,000.00   MATERIALITY:          50,000.00
                               PRECISION: 1,250,000.00    CONFIDENCE:95%
```

FROM	TO	FREQ	TOTAL	STD DEV	SAMP SIZE	PCT INT
1.00-	.01-	0	.00	.00	0	.0
.00	.00	2	.00	.00	0	.0
.01	12,703.33	953	5,001,351.63	3,587.93	44	4.6

12,703.34	19,693.75	313	5,005,174.04	2,068.55	8	2.5
19,693.76	26,366.66	221	5,020,718.42	1,956.58	6	2.7
26,366.67	32,816.66	171	5,028,150.72	1,833.99	4	2.3
32,816.67	39,825.00	138	5,020,074.35	2,024.71	4	2.8
39,825.01	50,000.00	109	4,847,630.16	2,938.04	4	3.6
50,000.01	64,800.00	63	3,503,127.19	3,756.94	63	100.0
FINAL TOTAL		1,970	33,426,226.51	14,449.26	133	6.7

THE RECOMMENDED SAMPLE SIZE LESS MATERIALITY 70

The STRATIF report lists the input parameters and the separation of the file into strata as defined by the stratum target size of 5,000,000. For each stratum the value of the smallest and largest item is listed along with the frequency, stratum total, and other statistics. The last stratum contains all records with values greater than the value specified for materiality (50,000). The ILSTRAT file contains the indicated number of records randomly selected from each stratum plus all records from the stratum of items greater than materiality.

The next job prints the CONFIRMATION-CONTROL report and the CONFIRMATIONS report. The CONFIRMATION-CONTROL report lists all appropriate fields for each record in the sample file. The CONFIRMATIONS report is formatted as a confirmation letter to be sent to all persons listed in the stratified sample file.

The following is the CONFIRMATION-CONTROL report. It lists the appropriate fields from each record in the stratified sample file.

```
                          LOANS SELECTED FOR POSITIVE CONFIRMATION                        PAGE      1
                           USING STRATIFIED MONETARY SAMPLING METHOD
                              AUDITED BANK OF BIG BANCSHARES

  ACCOUNT     LOAN                              LOAN ISSUE   MATURITY    ORIGINAL          LOAN
   NUMBER    NUMBER    BORROWER NAME               DATE        DATE       BALANCE         BALANCE

 00000500   00001077   CUSTOMER78                 5/31/86     5/31/91      1,000.00          250.00
                       7 MAIN ST. APT#26
                       ANYTOWN, ANYSTATE 63450

 00001354   00001931   CUSTOMER932                3/15/85     3/14/90     17,900.00          298.33
                       7 MAIN ST. APT#39
                       ANYTOWN, ANYSTATE 56072

 00001528   00002105   CUSTOMER1106               3/19/85     3/19/90     18,300.00          305.00
                       7 MAIN ST. APT#62
                       ANYTOWN, ANYSTATE 49357

 00001273   00001850   CUSTOMER851               10/11/87    10/10/90      1,600.00          355.55
                       7 MAIN ST. APT#18
                       ANYTOWN, ANYSTATE 66206

 00001823   00002400   CUSTOMER1401               3/09/87     3/08/90     19,900.00          552.78
                       7 MAIN ST. APT#64
                       ANYTOWN, ANYSTATE 72416

 00001820   00002397   CUSTOMER1398               4/26/85     4/25/90     18,400.00          613.33
                       7 MAIN ST. APT#79
                       ANYTOWN, ANYSTATE 97655

 00002235   00002812   CUSTOMER1813              12/13/89    12/13/90      1,000.00          833.33
```

```
                    7 MAIN ST. APT#34
                    ANYTOWN, ANYSTATE 89544

00002188  00002765  CUSTOMER1766          10/26/86   10/25/90      5,200.00         866.66
                    7 MAIN ST. APT#55
                    ANYTOWN, ANYSTATE 76069

00001774  00002351  CUSTOMER1352           2/27/88    2/26/95      1,700.00       1,214.28
                    7 MAIN ST. APT#42
                    ANYTOWN, ANYSTATE 32606

00001955  00002532  CUSTOMER1533          11/18/85   11/18/90      8,400.00       1,260.00
                    7 MAIN ST. APT#48
                    ANYTOWN, ANYSTATE 29052
   .        .        .                   .       .         .          .
   .        .        .                   .       .         .          .
   .        .        .                   .       .         .          .

00002131  00002708  CUSTOMER1709           2/16/90    2/16/92     64,800.00      64,800.00
                    7 MAIN ST. APT#13
                    ANYTOWN, ANYSTATE 29549

                                                                5601,200.00    4402,282.63
```

The final report is an example of one of the confirmation letters written by the CONFIRMATIONS report.

```
                 AUDITED BANK OF BIG BANCSHARES
                      1269 SUNRAY PARKWAY
                   SUNSHINE, FLORIDA  35060

                                             JULY 28, 1989

CUSTOMER 14
7 MAIN ST. APT#91
ANYTOWN, ANYSTATE 75606

GENTLEMEN:

WE ARE IN THE PROCESS OF AUDITING INSTALLMENT LOANS
AND WOULD LIKE FOR YOU TO VERIFY INFORMATION ABOUT
YOUR ACCOUNT. AS OF JULY 28, 1989 OUR RECORDS
SHOW THE INFORMATION LISTED BELOW. NOTE THAT THIS
CONFIRMATION IS ONLY FOR THE LOAN INDICATED, EVEN
THOUGH YOU MAY HAVE OTHER LOANS.

PLEASE INDICATE ON THE REVERSE SIDE OF THIS LETTER
WHETHER OR NOT YOU AGREE WITH THE INFORMATION LISTED
BELOW. IF YOUR RECORDS ARE DIFFERENT, PLEASE
INDICATE. RETURN THIS LETTER IN THE ENCLOSED,
POSTAGE PAID ENVELOPE.

     ACCOUNT      LOAN         LOAN        MATURITY
      NUMBER     NUMBER      BALANCE         DATE

     00000436   00001013      57,383.33    11/06/92

THANK YOU FOR YOUR COOPERATION.

                         SINCERELY,

                         AUDITED BANK OF BIG BANCSHARES
```

The REPORT statement uses parameters to control the format of the letter. NOADJUST specifies that all output is to be left justified on the page instead of being automatically centered. The lines of the letter are placed by specifying COL for each line. The NODATE and NOPAGE parameters suppress the printing of a date and page number while LINESIZE 70 specifies the length of each line of output.

The body of the letter is written by using TITLE lines. The account number, loan number, loan balance and maturity date are written with the standard LINE statement. The letter is completed by using LINE statements for the closing.

Job Summary

The following is a summary of the Case Study Three CA-PanAudit Plus routines and their associated parameters and files:

```
ROUTINE   PURPOSE           PARAMETERS                  FILES
CS3FILDF  file definition   none                        ILMAST
CS3FILGN  generate file     DATEVALUE, FORMAT, NUM-RECS  ILMAST
CS3JOB0   INTERVL analysis  none                        ILMAST
CS3JOB1   tasks A,B,C,D,E   DATEVALUE, FORMAT           ILMAST
CS3JOB2   task F            none                        ILMAST
CS3JOB3   task G            none                        ILMAST, ILATRIB
CS3JOB4   task H            none                        ILMAST, ILSTRAT
CS3JOBA   task A            DATEVALUE, FORMAT           ILMAST
CS3JOBB   task B            none                        ILMAST
CS3JOBC   task C            none                        ILMAST
CS3JOBD   task D            none                        ILMAST
CS3JOBE   task E            none                        ILMAST
CS3JOBF   task F            none                        ILMAST
CS3JOBG1  task G (sampling) none                        ILMAST, ILATRIB
CS3JOBG2  task G (report)   none                        ILMAST, ILATRIB
CS3JOBH1  task H (sampling) none                        ILMAST, ILSTRAT
CS3JOBH2  task H (report)   none                        ILMAST, ILSTRAT
```

Syntax

```
%CS3FILGN [DATEVALUE date] [FORMAT value] [NUM-RECS number]
```

`[DATEVALUE date]`

DATEVALUE controls the generation of dates in the ILMAST file. This involves generating dates in the past (such as loan date) and the future (such as maturity date). The default value is the current system date.

`[FORMAT value]`

Specify the format of DATEVALUE. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
```

Note: The only valid Julian format is YYDDD.

The default value for FORMAT is MMDDYY. In most cases, you only need to specify FORMAT if you specify DATEVALUE. However, if you allow DATEVALUE to default, and the current system date is not in the format MMDDYY, you must specify the proper format of the current system date.

`[NUM-RECS number]`

Specifies the number of records that CS3FILGN is to generate. The default value is 2201.

Syntax

```
%CS3JOB1 [DATEVALUE date] [FORMAT value]

                or

%CS3JOBA [DATEVALUE date] [FORMAT value]
```

`[DATEVALUE date]`

DATEVALUE controls the BASEDATE of the AGING routine. The default value is the current system date.

`FORMAT value`

Specify the format of DATEVALUE. This is a literal description of pairs of letters. The letters indicate positions as follows:

```
MM = month
DD = day
YY = year
CC = century
```

The following are valid formats:

```
MMDDYY
MMDDCCYY
YYMMDD
```

Note: The only valid Julian format is YYDDD.

The default value for FORMAT is MMDDYY. In most cases, you only need to specify FORMAT if you specify DATEVALUE. However, if you allow DATEVALUE to default, and the current system date is not in the format MMDDYY, you must specify the proper format of the current system date.

Operation

Allowing all three parameters in CS3FILGN to default produces results identical to those shown in the Case Study Three reports, with the exception of the exact values for the dates. The relationships between dates remain the same, as do all other numeric results. The generated dates are based on the date that you specify for DATEVALUE.

If DATEVALUE is allowed to default, the generated dates will be based on the current system date. Specify a DATEVALUE of 07/19/85 to produce the exact same results, including the dates, as shown in the Case Study Three reports.

In most cases, if you allow DATEVALUE to default when the file is generated, you can also allow DATEVALUE to default when jobs CS3JOB1 or CS3JOBA are run. However, if the file was generated on a previous day, jobs CS3JOB1 or CS3JOBA invoke AGING using the current system date for the BASEDATE. This creates a valid aging of the file from the current system date, but does not reproduce the same results as on previous days, because the BASEDATE for the analysis is defaulting to the current system date. This causes the age of a record to change with each subsequent day that you execute the routine. To maintain a consistent report, specify DATEVALUE as the date that the file was generated.

For the same reasons, to produce an AGING report identical to the previous listings, specify a DATEVALUE of 07/19/85 not only in CS3FILGN but also in CS3JOB1 and CS3JOBA.

In addition, task E uses the DAYSAGO routine, which calculates the elapsed number of days between the date that you specify and the current system date. This also causes differing results when you execute this task on any day other than the day that the file was created. Since there is no method available to specify a different base date for DAYSAGO, this routine always produces a report different than the Case Study Three report when it is not run on the day the file was created, or when you specify DATEVALUE in the file generation routine to change the base date.

The ILMAST file contains records 149 bytes long with a block length of 14900. The ILATRIB and ILSTRAT files contain fixed-length records of 149 bytes. CS3FILGN writes the generated records to the ILMAST file. CS3JOB3 and CS3JOBG1 create the sample file, ILATRIB, while CS3JOB4 and CS3JOBH1 create the sample file, ILSTRAT.

Index

%

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z