CA Culprit™ for CA IDMS™

Reference Guide

Release 18.5.00, 2nd Edition

CA Technologies Product References

This document references the following CA products:

- CA ADS™
- CA Culprit™ for CA IDMS™
- CA IDMS™/DB
- CA IDMS™ SQL
- CA IDMS™ Performance Monitor
- CA Librarian®
- CA Panvalet®

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Documentation Changes

The following documentation updates were made for the 18.5.00, 2nd Edition release of this documentation:

- IBM Language Environment Requirements (see page 15)—Added section with details on LE runtime requirements for z/OS and z/VSE.
- Overview (see page 29), One-Step and Five-Step JCL (see page 321)—Changed references to CULXPROF to CULPPROF.
- Update Installation Default Profile Options (see page 452)—Updated to cover only current 18.5.00, 2nd Edition functionality.
- z/VSE Operating System Procedures (see page 453)—Updated according to current 18.5.00, 2nd Edition functionality.
- Five-Step z/VSE JCL—Central Version (see page 342), Five-Step z/VSE JCL—Local Mode (see page 345)—Updated the example JCL to include SORT1.SAMPS instead of hard-coded sort parameters.
- The Operating System Option section was removed as it contained the documentation of the PROFILE OPSYS= parameter which is obsolete.
- The z/VM Operating System Procedures section was removed as it does not apply to release 18.5.

Contents

Chapter 9: CA Culprit in the CA IDMS/DB Environment 187

Chapter 1: Introduction

This manual is a reference for using CA Culprit.

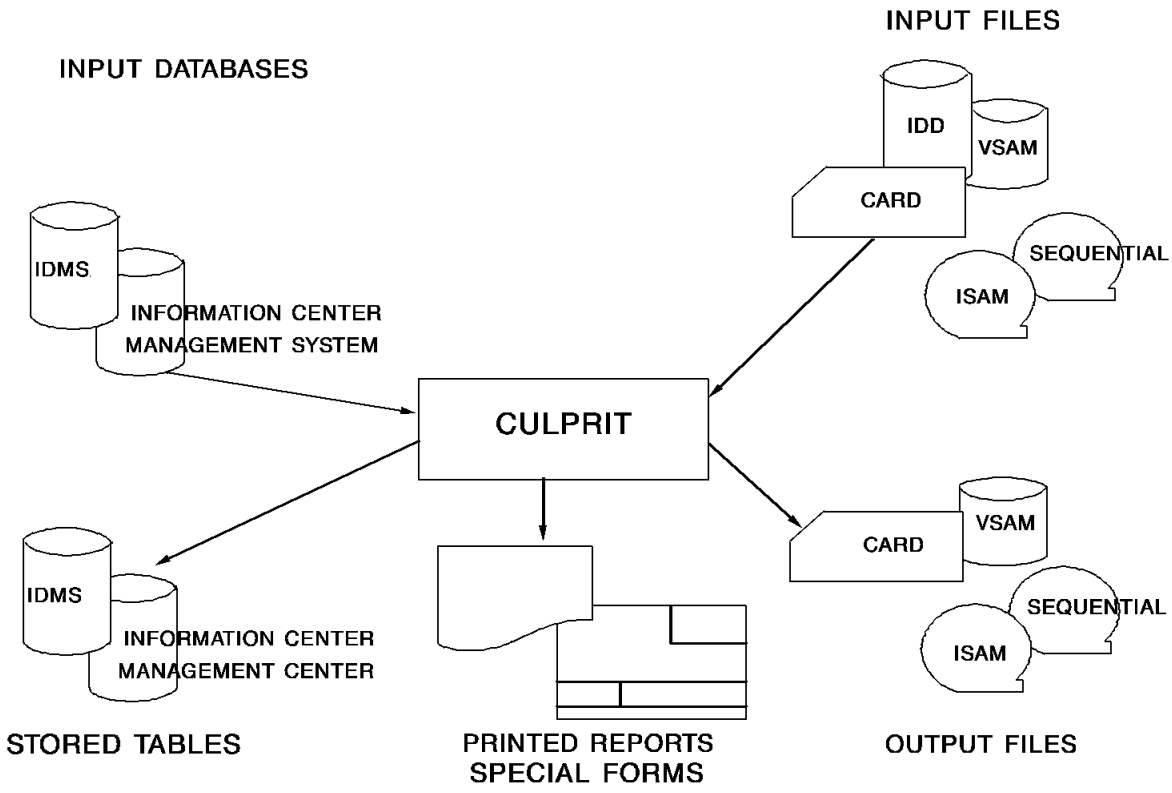
This section contains the following topics:

What It Does

CA Culprit is a batch utility that generates reports and tables from conventional and database files. CA Culprit is a batch utility that generates reports from conventional and database files.

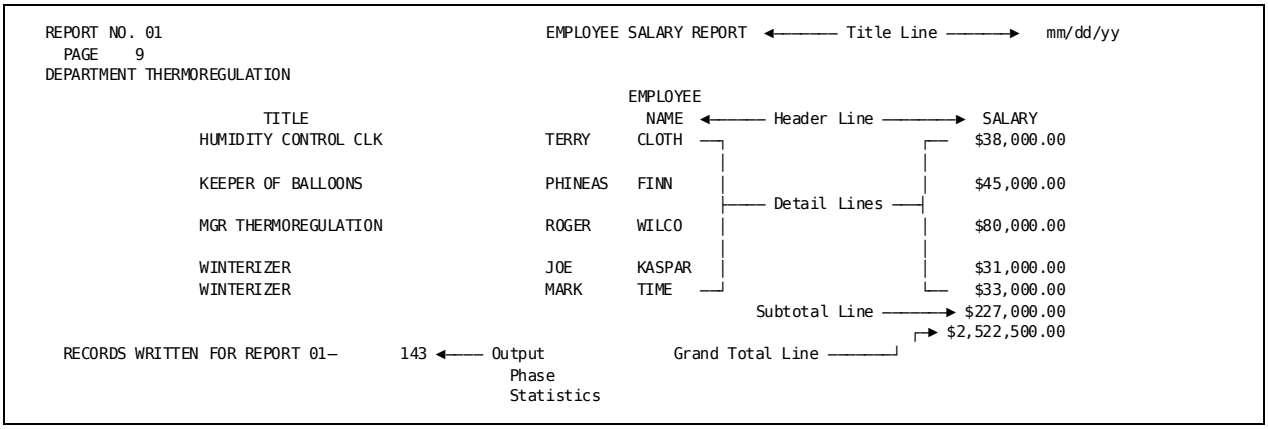
Data Input

CA Culprit can extract data from several database and conventional file structures as shown in the following figure. As many as 32 conventional files can be read and processed in one CA Culprit run.



Note: CA Culprit can extract data from and write reports to several database and file structures. CA Culprit is fully integrated with other products, such as CA IDMS/DB, the Integrated Data Dictionary (IDD), and CA-ICMS.

Each run can generate up to 100 reports from the same input data. Reports can be formatted as printed output, stored tables, or written to cards, tape, or disk. Following is an example of a printed CA Culprit report.



General CA Culprit concepts, coding considerations, and a description of the CA Culprit processing phases follow.

IBM Language Environment Requirements

Access to the IBM® Language Environment (LE) runtime support is required when accessing CA IDMS databases or dictionaries, and when calling LE programs.

Under z/OS, the LE runtime library (usually CEE.SCEERUN) must be in the STEPLIB concatenation or in the MVS linklist.

Under z/VSE, the LE runtime library (usually PRD2.SCEEBASE) must be in the LIBDEF PHASE search chain, temporary or permanent.

General Concepts

General CA Culprit concepts concern types of CA Culprit parameters, filling and processing the input buffer, and executing control breaks. Each topic is discussed separately below.

CA Culprit Parameters

The following table introduces each type of CA Culprit parameter and its function. It also indicates whether a parameter is **report specific** or **global**.

CA Culprit Parameter	Identifier	Status	Function
PROFILE	PROFILE	Global	Defines options to override system defaults
INPUT	INPUT	Global	Defines physical characteristics of the input file or files
REC	REC	Global	Defines characteristics of the input fields
SELECT/ BYPASS	SELECT or BYPASS	Global	Establishes criteria to select or bypass records to be processed
OUTPUT	OUTPUT	Report specific	Defines the physical characteristics of the output file
SORT	SORT	Report specific	Establishes an order for output records and defines process control breaks
Title	3 or T	Report specific	Specifies a title
Edit	4 or H 5 or D 6 or S	Report specific	Specifies the location and format of output fields on header, detail, and total lines
Process	7 or I 8 or B	Report specific	Specifies procedure logic for extract and output phase processing
Work field	0 or W 1	Global/ report specific	Defines a global or report-specific work field
Implied subscript	15	Global/ report specific	Provides an alternative name for a single occurrence of a multiply-occurring field and a means to increment subscript values for multiply-occurring fields
USE	USE	Not applicable	Copies stored code and modifies copied and inline code

CA Culprit Parameter	Identifier	Status	Function
=COPY	=COPY	Not applicable	Copies stored code, suppresses INPUT parameters, and changes report numbers on copied or inline code
=MACRO	=MACRO	Not applicable	Copies stored code and modifies copied and inline code

Report-specific Parameters

A report-specific parameter is available only to the report that defines the parameter. For example, an OUTPUT parameter associated with one report number can specify a printed report, while the OUTPUT parameter associated with another report number can specify a report written to tape. Report-specific parameters contain a report number in columns 2 and 3.

Global Parameters

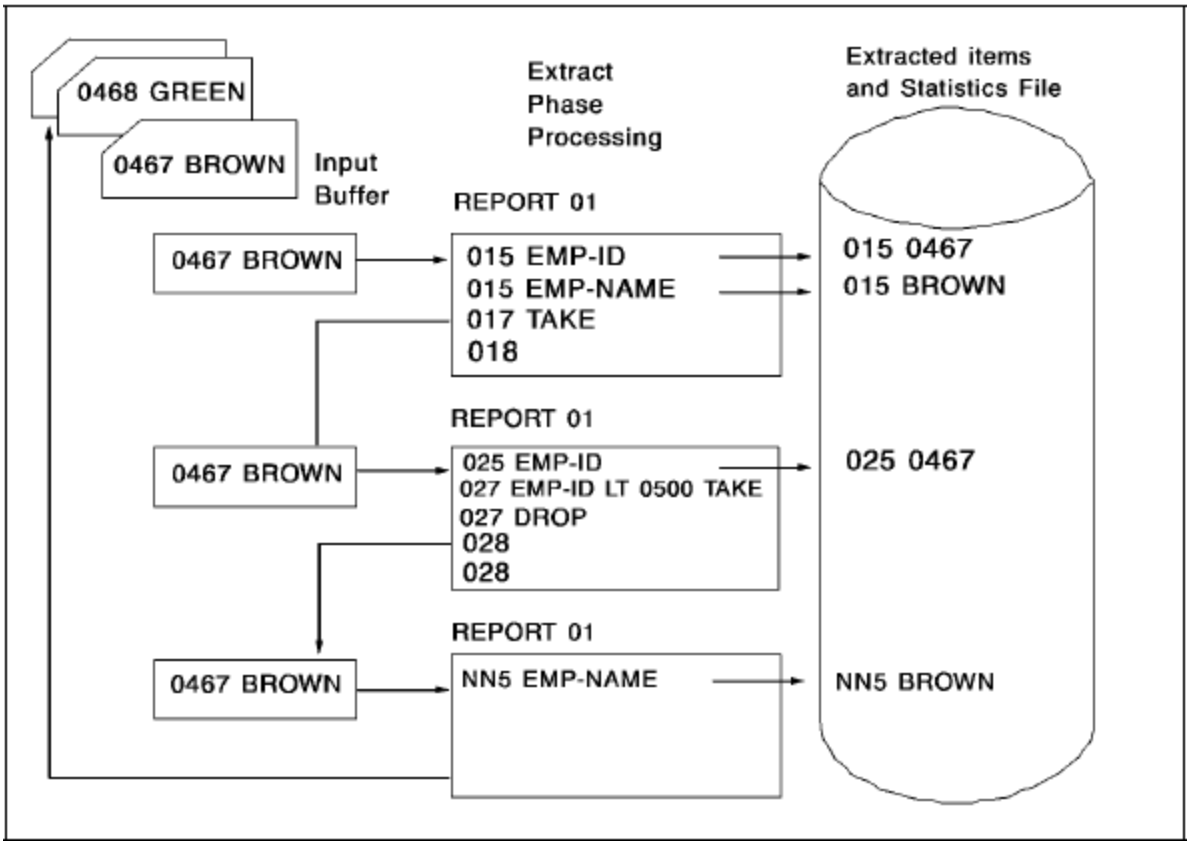
A global parameter is available to every report in the CA Culprit run. For example, the INPUT parameter defines the input file for the run, the REC parameter defines an input field for the run, and so on. Work field parameters can be defined as global or report specific. An example of a global work field is a constant that is referenced by more than one report in the run.

Input Buffer

CA Culprit places input data in an area called the input buffer. If a single input file is read, the buffer contains a single occurrence of an input record. If multiple input files are read, the buffer contains one record from each file.

During the extract phase of CA Culprit processing, the input buffer is processed by each report in the run, as shown in the following figure. Type 7 process parameters process the data contained in the input buffer; type 5 edit parameters extract the data contained in the input buffer. The extracted data is stored in an extracted items and statistics file.

After the input buffer passes through the type 7 logic of each report, it is filled with the next record or records and is processed by each report in the run. This cycle continues until an end-of-file condition occurs. At end-of-file, the extract phase is completed.



Note: The input buffer is processed by one report at a time; type 7 process parameters process the contents of the input buffer and type 5 edit parameters extract the contents of the input buffer to the extracted items and statistics file.

Control Break

A control break executes during the output phase and is specified on a SORT parameter in association with a sort-key field. When the value of the sort-key field changes, the control break executes. A control break also executes at the end of the output phase. When a control break executes, CA Culprit performs the following actions:

- Interrupts writing of detail lines specified by type 5 edit parameters
- Performs break processing defined by type 8 process parameters

- Writes total lines specified by type 6 edit parameters or, in the absence of type 6 edit parameters, writes automatic totals
- Writes header lines and a title line, if they are specified, at the top of each new report page

After the control break executes, CA Culprit resumes printing detail lines until the next control break is encountered.

Coding Considerations

CA Culprit parameters are coded in an 80-column card-image format. CA Culprit coding forms are available to aid the user in coding CA Culprit parameters.

General Rules

These rules apply to all CA Culprit parameters:

- All parameters must be coded in uppercase letters.
- All parameters, except the USE parameter, begin with a **fixed format** that identifies the parameter and continue with a **free format**. The length of the fixed-format portion differs for each parameter. In general, entries that appear in the free-format portion of the parameter must appear in the order specified by the parameter syntax; a comma or at least one space must separate one entry from another. Keyword expressions (for example, SZ=*field-size-n*) that follow positional entries can appear in any order.
- **Column 1** is always blank, except on a continuation line, a comment line, an =COPY parameter, an =MACRO parameter, or a USE parameter.

  One or more **continuation lines** can be coded for any CA Culprit parameter except the title parameter. For all CA Culprit parameters except USE, a continuation line is denoted by an asterisk (*) in column 1; the parameter continues in any column after column 1. A continuation line on a USE parameter can be coded starting in column 1; a special character is not required to indicate a continuation line.

  The continuation line must immediately follow the parameter it continues; the fixed-format portion of the parameter is not repeated on the continuation line. A keyword expression can span two lines, provided the equal sign (=) is not separated from the keyword. The example below illustrates a valid method to code a keyword expression:

```
 01510001 SALARY-OF-A-VICE-PRESIDENT F$ DP=2 SZ=
*20
```

- **Columns 73 through 80** are reserved for user sequence numbers. CA Culprit does not read columns 73 through 80; therefore, coded parameters must not extend beyond column 72.
- **Comments** are permitted on all CA Culprit parameters except the title parameter. Comments are introduced by a dollar sign ($); the dollar sign must be separated from the last coded entry on the parameter by at least one space or comma.

  Any number of comment lines can be added by coding the dollar sign ($) in column 1. The comment lines appear in the Sequential Parameter Listing but not in the Input Parameter Listing. Both listings are described later in this chapter.

Order of Parameters

CA Culprit parameters can be submitted in any order with the following exceptions:

- REC parameters must follow the INPUT parameter to which they apply and must precede the next INPUT parameter (if any).
- Process parameters without sequence numbers must be entered in the order they are to be executed.
- A SELECT/BYPASS parameter that specifies the keyword BUFFER must follow SELECT/BYPASS parameters that do not specify this keyword.

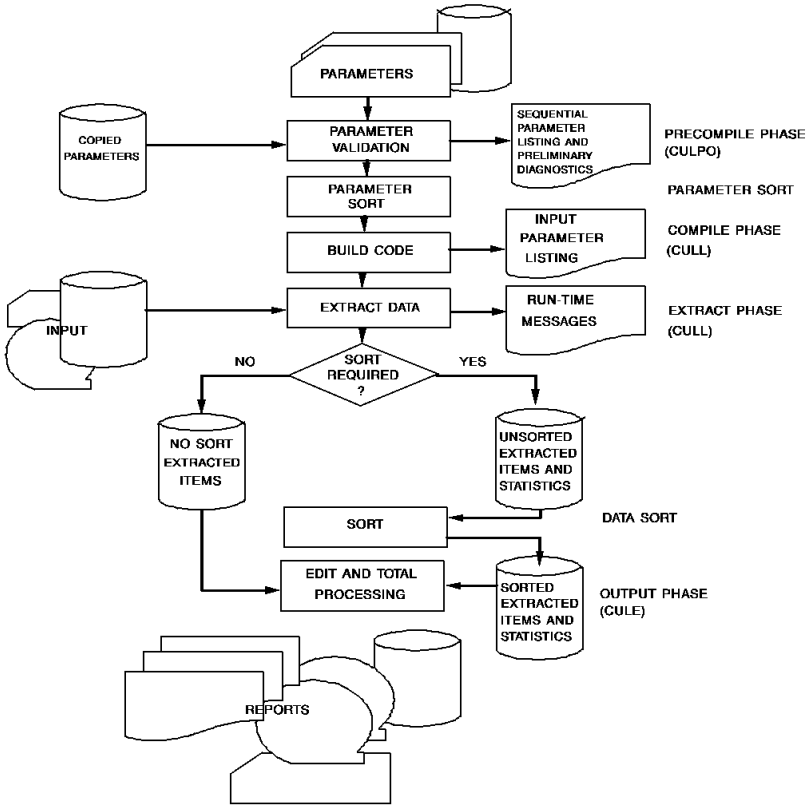
In general, parameters for single and match-file runs are submitted in the following order:

- PROFILE (optional)
- INPUT (required)
- REC (required)
- SELECT/BYPASS (optional)
- In match-file runs, repeat as required for each input file:
  - INPUT
  - REC
  - SELECT/BYPASS
- SELECT/BYPASS BUFFER (optional)
- Global work field and implied subscript parameters (optional)
- Report-specific parameters (repeat for each report in the CA Culprit run):
  - OUTPUT (optional)
  - SORT (optional)
  - Title (optional)

- Edit (at least one type 5 edit parameter is required)
- Process (optional)
- Work field/implied subscript (optional)

Processing Phases

CA Culprit processes information in six phases. The following figure illustrates these six processing phases. A description of each processing phase follows:



Note: CA Culprit's six processing phases are Precompile, Parameter Sort, Compile, Extract, Data Sort, and Output.

Precompile

The precompile (CULP0) phase performs preprocessing functions on CA Culprit input parameters:

- Copies parameters into the job stream as instructed on the USE, =COPY, and =MACRO parameters (see Chapter 8, **Copied Code Parameters**)
- Accumulates statistics that are used to determine resource requirements in subsequent processing phases
- Performs limited syntax verification on the input parameters

This phase generates the **Sequential Parameter Listing** as output. An example of a Sequential Parameter Listing appears in the following report. This listing shows the parameters that generated the output shown earlier; parameters appear in the order they were entered.

Sequential Parameter Listing for Employee Salary Report:

```
mm/dd/yy      SEQUENTIAL PARAMETER LISTING              Vnn.n  PAGE
00 ** SYSIN **                           IN  80 DD=EMPFILE
  INSTALLATION SECURITY OPTION IS NO
                                          REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
                                          REC EMP-LNAME  11 15
                                          REC SALARY     70 10 3 DP=2
                                          REC TITLE      50 20         'TITLE'
                                          REC DEPARTMENT 30 20         'DEPARTMENT'
                                          010 COUNT 1
                                          013EMPLOYEE SALARY REPORT
                                          01SORT DEPARTMENT,1,TITLE,0,EMP-LNAME
                                          01410010'DEPARTMENT'
                                          01410012 DEPARTMENT
                                          01510000 COUNT
                                          0151*010 TITLE HR   SZ=20
                                          0151*020 EMP-NAME HR
                                          0151*030 SALARY F$ SZ=10 HF
                                          0161*030 SALARY F$ SZ=10
                                          018010   IF LEVL EQ 1 DROP
```

Note: Parameters appear in the order in which they were entered by the user.

Parameter Sort

The parameter sort phase sequences all run parameters according to the CA Culprit standard sort; for example, parameters that share the same report number are ordered together. Run parameters include user-supplied parameters and parameters that are generated internally by CA Culprit during the precompile phase.

Compile

The compile (CULL) phase reads the sorted parameter file and generates a series of machine language subroutines for each report. If errors are detected during this phase, CA Culprit prints appropriate diagnostic messages; if the errors are severe enough, CA Culprit will not generate the associated reports.

This phase generates the **Input Parameter Listing** as output. An example of an Input Parameter Listing appears in the following report. This listing shows the parameters that generated the report shown earlier in **Data Input** under **What It Does**. The parameters are now sorted by type. The listing includes any default values supplied by CA Culprit to user-supplied parameters.

Input Parameter Listing for Employee Salary Report:

```
mm/dd/yy                  INPUT PARAMETER LISTING                Vnn.n  PAGE    1

********
INPUT    RECORD TYPE BLOCK  FILE DESCRIPTION...
********
 INPUT   00080   F
********
REC      START SIZE TYPE DP FIELD-NAME                    RECORD-NAME,LEVEL
********
 REC     00030  020         DEPARTMENT
 REC     00011  010         EMP-LNAME
 REC     00001  025         EMP-NAME                        'EMPLOYEE'     'NAME'
 REC     00070  010   3   2 SALARY
 REC     00050  020         TITLE
********
WORK     LENGTH   WORK-FIELD-NAME    OCCURRENCES  DECIMAL POINT AND VALUE
********
 01  0   008       COUNT     1
********
SORT     BREAK A/D  SORT FIELD-NAME
********
 01 SORT   1        DEPARTMENT
           0        TITLE
                    EMP-LNAME
********
TITLE    REPORT TITLE
********
 01  3   EMPLOYEE SALARY REPORT
********
EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
********
 01  4     1   0  0001   'DEPARTMENT'
 01  4     1      0012   DEPARTMENT
********
EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
********
 01  5     1      *010   TITLE     SZ=020    HF
 01  5     1      *020   EMP-NAME  HR
 01  5     1      *030   SALARY    SZ=010    F$        HF
 01  5     1      0000   COUNT
********
EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
********
 01  6     1      *030   SALARY    SZ=010    F$
********
PROCESS USER   INTERNAL
        LABEL  SEQUENCE     PROCESS STATEMENT
********
 01  8  010        1    $ IF LEVL EQ 1 DROP
 01  8             2    LEVL      EQ        1         DROP
 EXTRACT WILL BE PERFORMED                ┐
 PROFILE OPTION IN EFFECT: RELEASE =  6   ├──────────── Compile phase is successful;
                                          ┘             input files will be read.
```

Note: This listing shows the CA Culprit parameters after the precompile, parameter sort, and compile phases. Default values are supplied where applicable (for example, record type F on the INPUT parameter).

Extract

The extract (CULL) phase performs the following functions:

- Reads the input file or files and builds the input buffer
- Writes information extracted from the input buffer and stored in work fields to the extracted items and statistics file for use in the subsequent output phase

The following CA Culprit parameters participate in this processing phase:

- SELECT/BYPASS parameters evaluate each input record before it is placed in the input buffer.
- Type 5 edit parameters extract data contained in work fields and in each selected input buffer. The extracted data is stored in the extracted items and statistics file for use during the output phase.
- Type 7 process parameters process the data contained in each input buffer and work field.

If data from the input buffer is selected for a particular report, appropriate data is written to an extracted items file; if data from the input buffer is not selected for a particular report, control of the input buffer passes directly to the next report.

The extract phase generates the **Run-Time Messages** listing as output, as shown in the following report. This listing shows input processing statistics such as the number of records read and the number that passed through SELECT/BYPASS logic.

Run-Time Message Listing for Employee Salary Report:

```
mm/dd/yy                RUN TIME MESSAGES                    Vnn.n  PAGE

 **********  END OF FILE  ******************************************************
                                                                                 ├── Input file successfully read
         56 INPUT RECORDS READ                                                   │
                                                                                 └
```

Note: The END OF FILE message indicates that the input was successfully read.

Data Sort

The data sort phase performs the following functions:

- Sorts all of the items on the extracted items and statistics file built by the extract phase. The items are sorted for each report according to information coded on the SORT parameter.
- Returns the sorted items to the same file. The file contains the extracted and sorted input buffers and control records used by CA Culprit.

Input buffers and work fields that are extracted for a report but not sorted are written to an unsorted extracted items file. The unsorted extracted items file contains detail information for reports that either do not specify a SORT parameter or specify a SORT parameter that contains the keyword NOSORT. This file is not processed in the data sort phase.

Output

The output (CULE) phase performs the following functions:

- Totals numeric detail information contained in the extracted items and statistics file for each control break and at the end of the output phase.
- Performs logic defined by type 8 process parameters for each control break and at the end of the output phase.
- Generates reports line by line. As the lines are generated, they are printed or written to tape, cards, or disk, according to the information supplied on the OUTPUT parameter for a particular report.

The following CA Culprit parameters participate in the output phase:

- Type 3 title parameters write title lines at the top of each new page.
- Type 4 edit parameters write header lines at the top of each new page.
- Type 5 edit parameters write detail lines that contain information extracted from each input buffer.
- Type 6 edit parameters write total lines that can contain totals accumulated during the output phase, sort-key values established on the SORT parameter, and values established by type 8 procedure logic.
- Type 8 process parameters perform procedure logic.

This phase generates an **Output Phase Statistics** report at the end of each report, as shown earlier in **Data Input** under **What It Does**. The Output Phase Statistics report indicates the number of records written for the report.

Syntax Diagram Conventions

The syntax diagrams presented in this guide use the following notation conventions:

`UPPERCASE OR SPECIAL CHARACTERS`

Represents a required keyword, partial keyword, character, or symbol that must be entered completely as shown.

`lowercase`

Represents an optional keyword or partial keyword that, if used, must be entered completely as shown.

italicized lowercase

Represents a value that you supply.

lowercase bold

Represents a portion of the syntax shown in greater detail at the end of the syntax or elsewhere in the document.

Points to the default in a list of choices.

Indicates the beginning of a complete piece of syntax.

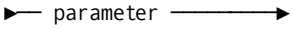
Indicates the end of a complete piece of syntax.

Indicates that the syntax continues on the next line.

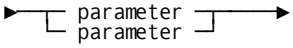
Indicates that the syntax continues on this line.

Indicates that the parameter continues on the next line.

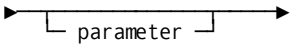
Indicates that a parameter continues on this line.



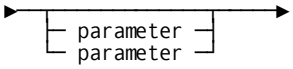
Indicates a required parameter.



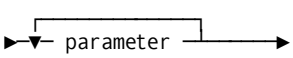
Indicates a choice of required parameters. You must select one.



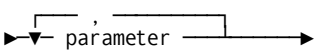
Indicates an optional parameter.



Indicates a choice of optional parameters. Select one or none.



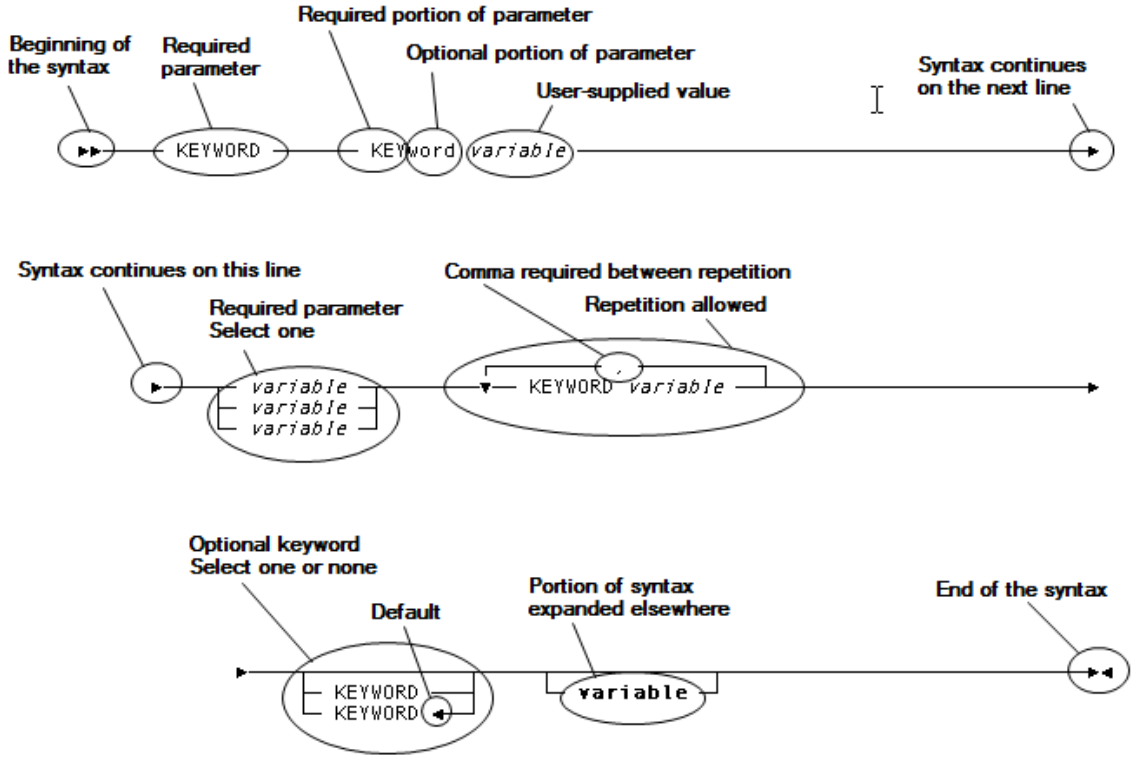
Indicates that you can repeat the parameter or specify more than one parameter.



Indicates that you must enter a comma between repetitions of the parameter.

Sample Syntax Diagram

The following sample explains how the notation conventions are used:



Chapter 2: PROFILE Parameter

This section contains the following topics:

More information:

PROFILE Parameter (see page 200)

Overview

What It Does

The PROFILE (PRO) parameter is an optional parameter that can be used to override system defaults on a run-by-run basis.

Note: The system profile established at CA Culprit installation may reflect different values from those indicated as defaults in this chapter. You may change the defaults for your site by altering the CULPPROF module. Refer to the appendix Profile Options (see page 441) for information on modifying CULPPROF.

Options specified on the PROFILE parameter can be grouped into three categories:

- Format options
- Site-specific options
- Error options

Format Options

Format options format and print CA Culprit parameter listings and reports. Examples of these actions are:

- The system time can be printed at the top of each page of output.
- European, Canadian, or American date formats can be requested.
- The number of characters per line and lines per page can be specified for the run.
- For special forms output, the first page of the report can be reprinted in order to properly align the form in the printer.
- Vertical or horizontal hexadecimal dumps can be requested.

Site-specific

Site-specific options perform installation-specific actions. Examples of these actions are:

- At z/OS installations with no relocating loaders, self-relocating user modules can be identified.
- CA Culprit processing modes can be specified to accommodate reports written for CA Culprit releases before 6.0.
- For CA Librarian users, a password code can be specified as needed.
- For Integrated Data Dictionary (IDD) users, a password code and user name can be specified as needed.
- One of four types of libraries can be identified that store CA Culprit parameters in library members; the stored parameters can be accessed through USE parameters, =COPY parameters, or =MACRO parameters.

Note: At installation time, users can select additional system profile options. For more information on establishing site-specific profile options, see the *CA IDMS Installation Guide* for your operating system.

Error Options

Error options specify actions in response to the severity level or the number of numeric errors generated during CA Culprit processing. Severity codes encountered as a result of a CA Culprit error are I (informational), W (warning), E (error), and F (fatal). Examples of these actions follow:

- CA Culprit can be instructed to generate reports containing I- and W-level errors even though other reports in the same run contain E- or F-level errors.
- At z/OS and z/VM installations, return code values can be set for each level of error; the values can be used to control the execution of subsequent job steps through execution JCL. At z/VSE installations, a nonzero return code forces job cancellation.
- CA Culprit outputs a buffer dump for every numeric error encountered during a processing phase up to a specified number. When the specified number of numeric errors is reached, CA Culprit can be directed to ignore additional errors, stop the CA Culprit run, or terminate the job step with an abend condition.

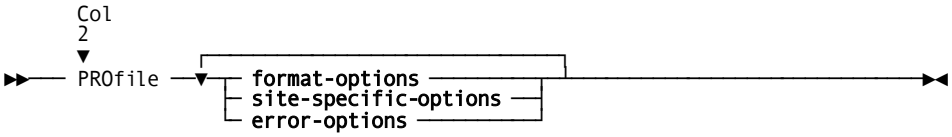
Note: For more information on error handling, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

PROFILE Syntax

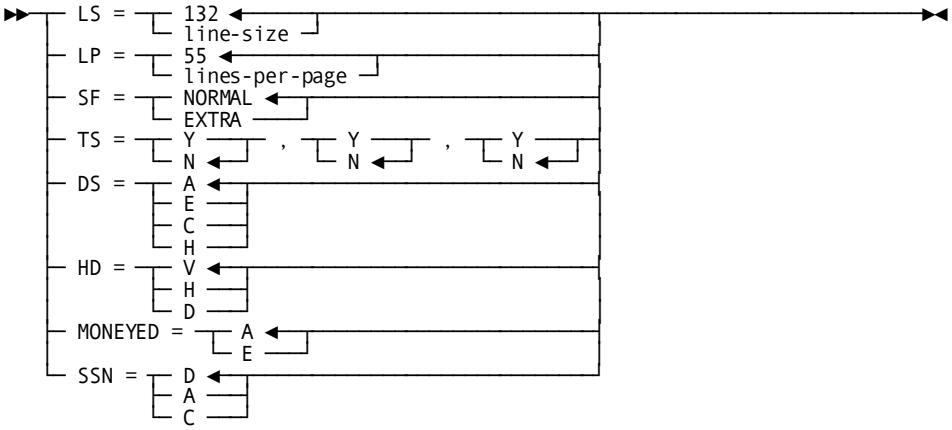
More information:

PROFILE Parameter (see page 200)

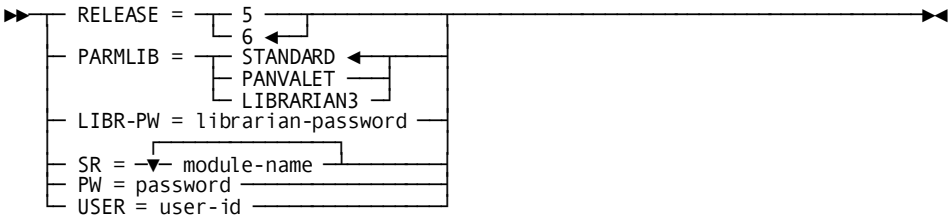
Overrides system defaults.



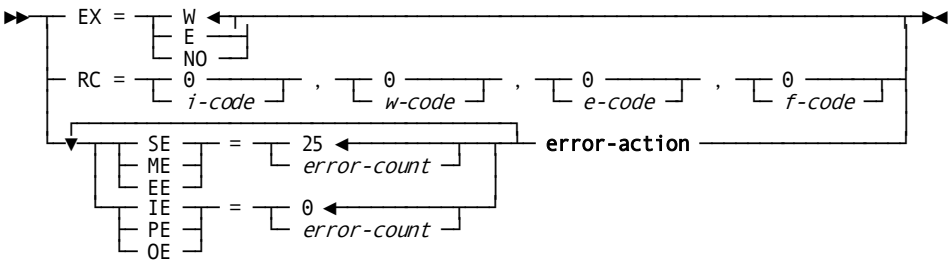
Expansion of Format-options



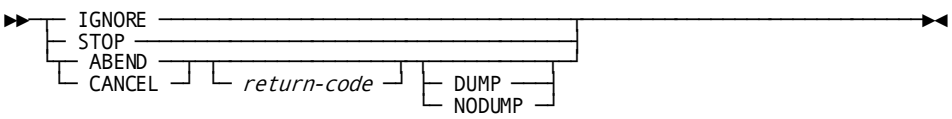
Expansion of Site-specific-options



Expansion of Error-options



Expansion of Error-action



PROFILE Parameters

PROfile

Specifies the parameter type. It must be coded starting in column 2.

Profile options can be coded in any column after PRO or PROFILE. Any number of options can be coded in any order. An option cannot be repeated.

format-options

Represents the expanded syntax explained below.

site-specific-options

For an explanation of **site-specific-options**, see the expanded syntax rules below.

error-options

For an explanation of **error-options**, see the expanded syntax rules below.

Note: For more information on numeric errors and the extended error-handling facility, see the *CA Culprit for Messages and Codes Guide*.

Expansion of Format-options

LS = 132/*line-size*

Specifies the line size for all printed reports in a CA Culprit run. Under z/OS and z/VM, at a z/VM installation, *line-size* must be in the range 1 through 150; under z/VSE, it must be in the range 1 through 132. The default is 132 characters for each line of output. *Line-size* can be changed for individual reports by coding *record-size* on the OUTPUT parameter.

LP = 55/*lines-per-page*

Specifies the maximum number of lines, excluding the title and headers, on each page of all reports in the CA Culprit run. The number must be in the range 1 through 32767. The default is 55 lines to a page. *Lines-per-page* can be changed for individual reports by coding the LP= specification on the OUTPUT parameter.

SF =

Refers to special forms output.

The SF= option is useful only when spooling is not in effect for printed output.

NORMAL

(Default) Indicates that the first page of output is not reprinted.

EXTRA

Indicates that the operator is to be prompted with a console message to mount special forms. The first page of output is reprinted as often as requested. This specification allows special forms to be properly aligned in the printer.

TS = Y/N,Y/N,Y/N

Indicates whether a time stamp, in *hh:mm* format, is to appear on CA Culprit parameter listings and output reports. If a time stamp is requested, it appears immediately after the system date.

The first indicator pair (Y/N) after the equal sign represents the time stamp on the Sequential Parameter Listing; the second indicator pair (Y/N) represents the time stamp on the Input Parameter Listing; the third indicator pair (Y/N) represents the time stamp on all titled user reports.

The default for each indicator pair is N; N specifies no time stamp. If one of the above indicators is specified, all three must be specified.

If a PROFILE parameter containing a TS= specification is the first CA Culprit parameter entered in a run, the time stamp appears on all pages of the Sequential Parameter Listing. If a PROFILE parameter containing a TS= keyword expression is not the first CA Culprit parameter entered in a run, the time stamp appears on every page except the first page of the Sequential Parameter Listing.

DS =

Specifies the format of the date stamp that appears on all CA Culprit parameter listings and output reports.

A

(Default) Specifies the American standard date format *mm/dd/yy*.

E

Specifies the European standard date format *dd/mm/yy*.

C

Specifies the Canadian date format *yy/mm/dd*.

H

Specifies the hajreh (Arabic) date format.

HD =

Specifies the format of hexadecimal dumps produced by the extended error-handling facility.

V

(Default) Specifies a vertical dump format. Alphanumeric characters print above the zone and digit portions of each character.

H

Specifies a horizontal dump format. The zone and digit portions of each character alternate across a line on the left side of the output page; the alphanumeric characters print on the right side of the output page.

Note: The following report shows sample horizontal and vertical hexadecimal dumps. For more information on the extended error-handling facility, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

D

Generates dumps in a vertical format using a DBCS character set.

```
mm/dd/yy                   RUN TIME MESSAGES               Vnn.n  PAGE    1

  INVALID NUMERIC DATA ENCOUNTERED IN REPORT 01
  ERROR OCCURRED DURING PROCESSING DETAIL   LINE 1 FIELD   3
  MOST RECENT SUBSCRIPT VALUE           0
  INPUT BUFFER NUMBER           1
  RECORD BUFFER DUMP
              CHAR   TERRY     CLOTH              THERMOREGULATION    HUMIDITY CONTROL CLK
              ZONE   ECDDE44444CDDEC4444444444444ECCDDDDCCEDCECDD4444CEDCCCEE4CDDEDDD4CDD00000030004
              DIGIT  359980000033638000000000000038594695743139650000844949380365396303320000008000C0
                    01...5...10....5...20....5...30....5...40....5...50....5...60....5...70....5...80



mm/dd/yy                   RUN TIME MESSAGES               Vnn.n  PAGE    1

  INVALID NUMERIC DATA ENCOUNTERED IN REPORT 01
  ERROR OCCURRED DURING PROCESSING DETAIL   LINE 1 FIELD   3
  MOST RECENT SUBSCRIPT VALUE           0
  INPUT BUFFER NUMBER           1
  RECORD BUFFER DUMP
POSITION  ADDRESS  STORAGE

 00001    12A7C0   E3C5D9D9 E8404040 4040C3D3 D6E3C840   40404040 40404040 40404040 40E3C8C5   *TERRY     CLOTH              THE*
 00033    000020   D9D4D6D9 C5C7E4D3 C1E3C9D6 D5404040   40C8E4D4 C9C4C9E3 E840C3D6 D5E3D9D6   *RMOREGULATION    HUMIDITY CONTRO*
 00065    000040   D340C3D3 D2000000 00000038 00000C40                                        *L CLK.........                 *
```

MONEYED =

Indicates American or European numeric editing.

A

(Default) Specifies that default numeric editing is to be done according to the American standard, where decimal digits are separated from integer digits with a period, and integer digits are separated into groups of three with commas. For example, 1,000.52.

E

Specifies that default numeric editing is to be done according to the European standard, where decimal digits are separated from integer digits with a comma, and integer digits are separated into groups of three with a period. For example, 1.000,52.

SSN =

Specifies the default format in which social security numbers will appear when using an FS edit mask.

D

Formats social security numbers using the date stamp (DS) option. This is provided for upward compatibility.

A

Formats social security numbers in American format: nnn-nn-nnnn.

C

Formats social security numbers in Canadian format: nnn-nnn-nnn.

Default: D

Expansion of Site-specific-options

RELEASE = 5/6

Indicates the CA Culprit processing mode under which the program is to run, as described below.

RELEASE = 5

Insures that programs written and previously run under Release 5.0 produce the same results when run under CA Culprit Releases 6.0 and later.

RELEASE = 6

(Default) Uses the full capabilities of the most current release of CA Culprit (Release 6.0 and later). Programs written and previously run under CA Culprit Release 5.0 may yield different results when executed under RELEASE=6.

PARMLIB =

Specifies the source library accessed for all reports in a run.

If this specification is omitted from a run that contains a USE parameter, =COPY parameter, or =MACRO parameter, source code is retrieved from the library member named in the PARMLIB= keyword expression during the last assembly of the CULL-PROF macro.

Note: For more information on the CULL-PROF macro, see the *Installation Guide* for your operating system.

STANDARD

(Default) Indicates that the source is retrieved from a z/VSE source statement library (SSL), a z/OS partitioned data set (PDS) source library; the source can be retrieved from a z/VM MACLIB or a z/OS PDS.

PANVALET

Indicates that the source is retrieved from the CA Panvalet library.

LIBRARIAN3

Indicates that the source is retrieved from a CA Librarian Release 3.0 or greater library.

LIBR-PW = *librarian-password*

Specifies the management (MCD) code of the master file. It is required to access modules in PROD2 status. This code is not printed in any CA Culprit listings.

SR = *module-name*

(z/OS only) Enables users without a relocating loader to identify self-relocating user modules so that CA Culprit can load those modules at the most advantageous core location.

Module-name must be the name used in the linkage editor PHASE statement when the module was link edited. Module names specified by the SR= option are added to the list of self-relocating CA-supplied and user-coded modules that is established at installation time. As many as 12 self-relocating modules can be specified by the user.

PW = *password*

(IDD users only) Specifies the security password associated with a user defined to IDD. This PROFILE parameter option is required when IDD security features are in effect; the password is not printed in any CA Culprit listings.

USER = *user-id*

(IDD users only) Specifies the name of a user defined in IDD. This PROFILE parameter option is required when IDD security features are in effect. The PROFILE parameter must appear before the INPUT parameter if USER is required because the user is validated immediately after the INPUT parameter is read.

Expansion of Error-options

EX =

Specifies the error conditions under which CA Culprit is to read and report on one or more user input files.

Note: For more information on severity codes associated with processing errors, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

W

(Default) Indicates that reports are produced only if no E- or F-level errors are encountered in the run. If E- or F-levels are encountered during verification of the input parameters, input file processing is canceled. The compile phase is completed to allow input parameter validation, but the input file is not read.

E

Indicates that E-level errors can occur without canceling the extract phase. Reports that contain no errors and reports that contain only W-level errors are generated. Reports that contain E-level errors are not generated. If any report contains an F-level error, the extract phase is terminated.

NO

Indicates that the input files are not read and that no reports are generated. This option allows input parameter validation only.

RC = 0/*i-code*,0/*w-code*,0/*e-code*,0/*f-code*

Specifies four return codes associated with four severity codes: I, W, E, and F, respectively. Each return code must be in the range 0 through 4095. The default for each code is 0. However, if one return code is specified, all four codes must be explicitly stated even if the default value is taken.

When CA Culprit executes as a five-step job, a return code specification affects the job execution of each operating system:

- Under z/OS and z/VM, the return code associated with the most severe error level is the return code for that step. The return code can be used to control the execution of subsequent steps through execution JCL.
- Under z/VSE, a nonzero return code causes the run to cancel after completing the step in which the indicated error level was encountered.

When CA Culprit executes as a one-step job in z/OS, z/VSE, and z/VM environments, a nonzero return code specification affects the job execution:

- In the precompile phase, the run terminates at the end of the precompile phase.
- In the compile or extract phase, the run terminates at the end of the extract phase.
- In the output phase, the run terminates at the end of the output phase.

  Note: (z/OS, z/VM, and z/VSE users) If a nonzero return code is issued for I-level and W-level messages, the job will be canceled since CA Culprit always issues informational messages and often issues W-level messages.

SE = 25/*error-count*

Specifies the maximum number of numeric errors to be reported by the extended error-handling facility during SELECT/BYPASS parameter processing. *Error-count* must be a number in the range 0 through 32767; the default is 25.

ME = 25/*error-count*

Specifies the maximum number of numeric errors to be reported by the extended error-handling facility during processing of match keys in a match-file run. *Error-count* must be a number in the range 0 through 32767; the default is 25.

EE = 25/*error-count*

Specifies the maximum number of numeric errors to be reported by the extended error-handling facility during the extract phase of processing. *Error-count* must be a number in the range 0 through 32767; the default is 25.

IE = 0/*error-count*

Specifies the maximum number of numeric errors reported by the extended error-handling facility during execution of a user input module. *Error-count* must be a number in the range 0 through 32767; the default is 0.

PE = 0/*error-count*

Specifies the maximum number of numeric errors reported by the extended error-handling facility during execution of a user procedure module. *Error-count* must be a number in the range 0 through 32767; the default is 0.

OE = 0/*error-count*

Specifies the maximum number of numeric errors reported by the extended error-handling facility during the output phase of processing, including execution of a user output module (if any). *Error-count* must be a number in the range 0 through 32767; the default is 0.

error-action

Indicates the action to be taken in the event that one of the error thresholds described above is exceeded.

The expanded syntax is explained below.

Expansion of Error-action

IGNORE

Allows processing to continue after the specified limit is exceeded; additional errors are not reported. The end-of-file statistics show the total number of errors encountered. IGNORE is the default action for SE=, ME=, and EE= specifications.

STOP

Forces end-of-file processing when the specified limit is exceeded. For each keyword expression described above except OE=, STOP has the same effect as a STOP-RUN statement in type 7 or type 8 procedure logic. When STOP is specified for OE=, only the current report is terminated.

ABEND

Causes abnormal termination of the CA Culprit phase if the specified limit is exceeded. ABEND is the default action for IE=, PE=, and OE= specifications. CANCEL may be used as a synonym for ABEND.

return-code

(z/OS only) Is an optional value that specifies the abend condition.

Return-code must be a number in the range 0 through 4095. The default is 0 for SE=, ME=, and EE= specifications; it is 999 for IE=, PE=, and OE= specifications.

DUMP/NODUMP

Causes a dump for the abend condition; NODUMP suppresses a dump. NODUMP is the default for SE=, ME=, and EE= specifications; DUMP is the default for IE=, PE=, and OE= specifications.

Examples

Sample PROFILE parameters are shown and described below.

Example 1

```
PROFILE HD=H LS=70 LP=40
```

All reports associated with this PROFILE parameter have a maximum of 40 lines (excluding the title and headers, if any) for every page of output. Each line has 70 characters. Any hexadecimal dumps that are generated by the extended error-handling facility have a horizontal format.

Example 2

```
PROFILE SE=25,ABEND,12,DUMP RC=0,0,0,16
```

If more than 25 numeric errors are encountered during SELECT/BYPASS processing, the job step terminates. When the step terminates, CA Culprit generates a return code of 12 and produces a dump. Although 25 errors is the default value, it must be specified on this PROFILE parameter because subsequent defaults within the expression are changed (that is, ABEND overrides IGNORE, 12 overrides 0, and DUMP overrides NODUMP).

If CA Culprit encounters F-level errors, it generates a return code of 16. In this example, return codes for I-, W-, and E-level messages must be explicitly stated because a nonzero return code for F-level errors is requested.

At a z/VSE installation, the job cancels at the end of the job phase in which CA Culprit encounters an F-level error. At z/OS and z/VM installations running five-step CA Culprit jobs, the return code can be coded on JCL statements to control the execution of subsequent CA Culprit steps.

Example 3

```
PROFILE EX=E RC=0,4,8,12
```

Reports that are error-free are generated regardless of the occurrence of E-level errors in other reports. Reports with I- and W-level errors only are considered to be error free.

One return code is specified for each error severity level; that is, 0 for I-level messages, 4 for W-level messages, 8 for E-level messages, and 12 for F-level messages.

At z/OS and z/VM installations, if I-, W-, and E-level messages are detected in a single processing phase, a code of 8 (the highest applicable code) is returned for that step. At a z/VSE installation, a W-level error cancels the job at the end of the step in which the error is detected. This action occurs because the PROFILE specifies a nonzero return code for W-level errors.

Example 4

```
 PROFILE DS=E  EE=50  TS=N,N,Y
*   $THESE REPORTS ARE FOR DISTRIBUTION TO ALL EUROPEAN AFFILIATES
```

In this example, as many as 50 numeric errors that occur during the extract phase are reported; any additional numeric errors that occur during this phase are ignored.

The date appears on parameter listings and reports in the European format, *dd/mm/yy*. The time stamp appears on all titled reports but not on parameter listings. A continuation line is coded to accommodate a long comment.

Example 5

```
PROFILE USER=CLK PW=PERU
```

For IDD users, a user name and password are submitted to IDD in order to access files defined to the data dictionary.

Example 6

```
PROFILE PARMLIB=LIBRARIAN3
```

This example specifies that parameters copied by a USE parameter, =COPY parameter, or =MACRO parameter are to be copied from a CA Librarian library.

More information:

Process Parameters (see page 103)
Release 5 and 6 Default Actions (see page 365)
Output Definition Parameters (see page 69)
Printing Parameter Lists (see page 305)
Copied Code Parameters (see page 149)
Execution JCL (see page 315)

Chapter 3: Input Definition Parameters

This section contains the following topics:

Overview

Input definition parameters describe input files and records and define conditions for selecting input records for processing. The three types of input definition parameters are listed below:

- **INPUT (IN) parameters** define the physical characteristics of input files.
- **REC parameters** define the physical characteristics of data fields in the input files.
- **SELECT/BYPASS (SEL/BYP) parameters** establish selection criteria that are applied to input records.

Each type of input definition parameter is discussed separately below, followed by examples of coding all three types of input definition parameters.

INPUT Parameter

More information:

INPUT Parameter (see page 203)

INPUT Parameter (see page 295)

Purpose

Defines the general characteristics of each input file to be read.

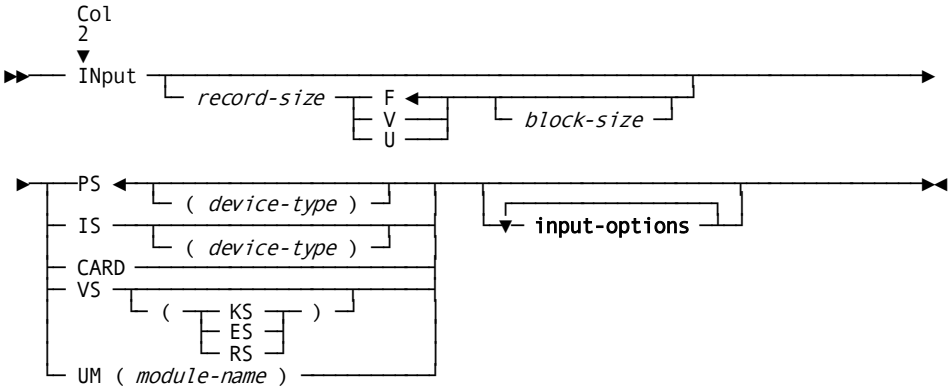
The input file can be a sequential tape or disk file, indexed sequential disk file, card file, or VSAM file. The file can also be defined to IDD. One INPUT (IN) parameter is required for each file read. CA Culprit can read and process up to 32 files per run.

During processing, CA Culprit places input data in a contiguous area called the **input buffer**. If a single input file is read, the buffer contains a single occurrence of the input record; when multiple input files are read, the buffer contains one record from each file.

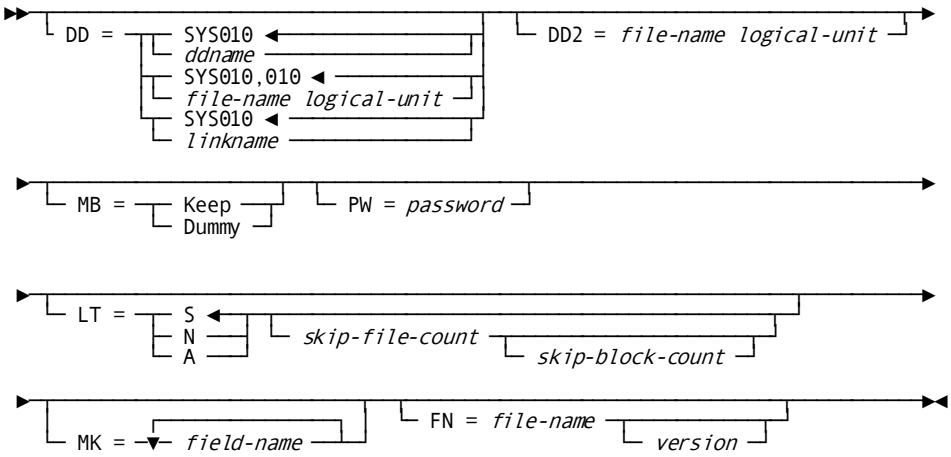
In a match-file run, CA Culprit establishes a status byte for each input file (a file-specific status byte) and a composite indicator (M*ID) that reflects the status of any or all input files. The file-specific status byte appears in the input buffer following each input record; the composite indicator appears in the last byte position of the input buffer.

Note: For more information on match-file runs, see the *CA Culprit for CA IDMS User Guide*.

Syntax



Expansion of Input-options



Syntax Rules

INput

Specifies the parameter type. It must be coded starting in column 2.

record-size

Specifies the size of the input record. *Record-size* must be a number in the range 1 through 32767; this value is coded in any column after IN or INPUT. This value is required for all file types (see below) except CARD.

For variable-length records, *record-size* is the length of the longest record in the file; this value must include the 4-byte record descriptor word.

F

Specifies that the input file contains fixed-length records. F is the default.

V

Specifies that the input file contains variable-length records.

U

Specifies that the length of the records in the input file is undefined.

block-size

Specifies the size of a physical block of records. This value must be a number in the range 1 through 32767. The following considerations apply to *block-size*:

- Under z/VSE, *block-size* is required when the file type is PS or IS (see below).
- Under z/OS and z/VM, *block-size* is optional.
- For fixed-length records, *block-size* must be an even multiple of *record-size*.
- For variable-length records, *block-size* must equal the longest physical block in the input file and must include the 4-byte block control prefix.
- For records of undefined length, *block-size* must be the same as *record-size*.

PS (*device-type*)

Specifies sequential file organization. This is the default. *Device-type* is an optional qualifier that must be enclosed in parentheses. Possible values for *device-type* are TAPE, 2311, 2314, 3310, 3330, 3340, 3350, 3370, 3375, 3380, and FBA.

IS (*device-type*)

Specifies indexed sequential file organization. *Device-type* is an optional qualifier that must be enclosed in parentheses. Possible values for *device-type* are 2311, 2314, 3310, 3330, 3340, 3350, 3370, 3375, 3380, and FBA.

Note: Under z/OS and z/VM, users can specify *device-type* for documentation purposes. Under z/VSE, CA Culprit determines the device type if the file is assigned to a particular device in execution JCL. *Device-type* must be specified on the INPUT parameter or assigned in execution JCL. This rule applies even for a tape or disk management system (for example, TFAST, DFAST) that ignores the assignment.

CARD

Specifies an input card file. If CARD is specified, all other INPUT parameter specifications can be omitted.

VS

(z/OS, z/VSE, and z/VM only) Indicates a virtual storage (VSAM) file organization and can be qualified in one of three ways.

If one of the qualifiers is specified, it must be enclosed in parentheses; for example, VS(RS).

KS

Indicates a key-sequenced VSAM file. This is the default.

ES

Indicates an entry-sequenced VSAM file.

RS

Indicates a relative-record VSAM file.

UM (*module-name*)

Specifies a user input module. The module name must be enclosed in parentheses, and is specified as follows:

- Under z/VSE, *module-name* must be the same as the name used on the linkage editor PHASE card when the module was link edited.
- Under z/OS, *module-name* must be the name or alias of the load module.

Note: For more information on user input modules, see the *CA Culprit for CA IDMS User Modules Guide*.

input-options

See the expanded syntax below.

DD =

Specifies the ddname (z/OS and z/VM) or filename (z/VSE) for the input file.

In a match-file run, the second and subsequent input files default to SYS011, SYS012, and so forth. If DD= is specified, it must appear on the INPUT parameters for each file in the run. These files must be defined in the execution JCL.

Note: *Ddname*, *file-name* and *logical-unit*, or *linkname-a* specifications must not conflict with CA Culprit system file assignments; these assignments can be changed, as described in the *CA IDMS Installation Guide* for your operating system.

SYS010/*ddname*

(z/OS and z/VM only) Specifies the name assigned to the DD statement that defines the input file; the default is SYS010.

SYS010,010/*file-name logical-unit*

(z/VSE only) Specifies the name assigned to the input file in execution JCL and the number of the device on which the appropriate file resides; the default filename is SYS010.

Logical-unit must be a number in the range 0 through 255. When DD=*file-name* is specified, *logical-unit* must also be specified. The default logical unit number is SYS010.

z/VSE users can use system standard label information as an alternative to inserting input file TLBL or DLBL and EXTENT JCL statements in the CA Culprit job stream. If filenames and logical unit numbers used in the standard label information area differ from CA Culprit input file requirements, then a DD= specification can be used to modify CA Culprit requirements, as shown in the following table.

SYS010/linkname

(BS2000/OSD only) Specifies the name associated with the input file by means of the /ADD-FILE-LINK command; the default is SYS010.

System Standard Label Information	DD= Statement	Explanation
// TLBL SYS010,'TAPE-DATA' // ASSGN SYS010,X'282'	None required	The label information agrees with the CA Culprit input file default, so no specification is required.
// TLBL SYS025,'TAPE-DATA' // ASSGN SYS010,X'282'	DD=SYS025,010	The specification uses the same filename as in the standard label information (SYS025) and specifies the default logical unit number (010). SYS010 must be assigned to the appropriate tape device.
// TLBL TAPEIN,'TAPE-DATA' // ASSGN SYS015,X'282'	DD=TAPEIN,15	The specification uses the same filename as in the standard label information (TAPEIN) and indicates a logical unit number of 015. SYS015 is assigned to the appropriate tape device.

System Standard Label Information	DD= Statement	Explanation

// DLBL MASTER,'DISK-FILE',0
// EXTENT SYS020,111111,,,19,380
// ASSGN SYS020,DISK,
VOL=PRODISK,SHR
	DD=MASTER,20	The specification uses the same filename as on the DLBL statement and the same logical unit number as on the EXTENT statement. SYS020 is assigned to the appropriate disk device.

Note: The first three table entries are tape label examples; the fourth entry is a disk label example. Assignments can be either permanent or made at run time.

DD2 = ***file-name logical-unit***

(z/VSE only) Specifies an alternating assignment for sequential tape or disk files. At run time, CA Culprit treats an end-of-file condition as if it were an end-of-volume condition; CA Culprit can therefore read multiple files of the same format as though they were part of a single file. The file type for these files must be PS.

File-name is the name of the alternating file. *Logical-unit* is a number in the range 0 through 255 that specifies the logical unit number.

Additional considerations for tape and disk files follow:

- For tape files, *file-name* and *logical-unit* alternate between the primary file (that is, the file defined in the DD= specification or the default) and the file identified in the DD2= keyword expression.
- For disk files, CA Culprit reads the first file from the file defined in the DD= specification or the default; it reads the second file from the *file-name* and *logical-unit* defined in the DD2= specification. File information for any subsequent files is obtained by adding 1 to both the *file-name* and the *logical-unit* defined in the DD2= specification; *file-name* must be specified in the form SYS*nnn* where *nnn* is a 3-digit number in the range 000 through 999.

In response to a DD2= specification, CA Culprit prompts the operator for additional input at end-of-file for each file. If the operator's reply is N or n, normal end-of-file procedures apply; if the reply is Y or y, CA Culprit reads another file, as described above. CA Culprit performs label processing and special positioning for each file if these operations are requested.

Under z/OS and z/VM, CA Culprit performs alternate file processing (concatenation) through execution JCL.

MB

Specifies input buffer options as follows:

Keep

Applies to match-file runs. When MB=KEEP is specified, CA Culprit retains a record in the buffer until a new record from the same file replaces it.

Note: For more information on match-file runs, see the *CA Culprit for CA IDMS User Guide*.

Dummy

Specifies that a portion of the input buffer is to be reserved for special handling. The value of *record-size* on this INPUT parameter determines the size of the dummy area. Subsequent REC parameters define specific areas within the dummy area, as required by the CA Culprit program. Space is reserved in the input buffer according to where the MB=DUMMY request is encountered in the input stream.

Dummy buffer areas can be used by user procedure modules (see the *CA Culprit for CA IDMS User Modules Guide*) or to establish a storage area that is available to all reports in a CA Culprit run.

PW = *password*

A unique 1- to 8-character password that can be inspected by a user input module.

If special characters are used (that is, characters other than letters, numbers, or hyphens), the password must be enclosed in single quotation marks. Hexadecimal literals must appear in the form X'*password-string*'.

LT =

Specifies label type information for PS (sequential) files:

S

(Default) Indicates standard labels.

N

Indicates that labels are omitted on a tape file. Under z/VSE, users can optionally specify a number of files and blocks of data to be skipped over when reading the tape.

skip-file-count

Indicates the number of files (tape marks) to be skipped;

skip-block-count

Indicates the number of blocks of data at the beginning of the file that are to be skipped. Both numbers must be in the range 0 through 32767.

A

Indicates standard and user-defined labels. This specification is invalid for match-file runs.

MK = *field-name*

Specifies from one to four match keys for match-file processing. *Field-name* is discussed under REC Parameter?Overview.

Note: For more information on match-file runs, see the *CA Culprit for CA IDMS User Guide*.

FN =

Identifies a nondatabase file defined to IDD:

If this keyword expression is specified and appropriate values are stored in the Integrated Data Dictionary (IDD), IDD automatically supplies the file record size, record type, block size, file type, device type, or user module name and label type. IDD also automatically generates a REC parameter for every field referenced on other CA Culprit parameters if the field is in a record in the file.

If other specifications appear on an INPUT parameter that specifies FN=*file-name*, these specifications override values stored in IDD unless CA Culprit security is in effect.

file-name

The 1- to 32-character name of a file whose field definitions are to be used in subsequent CA Culprit parameters. If the file name contains anything other than letters, numbers, or hyphens, it must be enclosed in quotation marks.

version

Identifies the file version being accessed; the default is the highest existing version number for the named file.

Examples

Sample INPUT parameters are shown and described below.

Example 1

```
INPUT  80  DD=INPUT1
```

In a z/OS environment, input is a sequential file that contains 80-byte records. The file is identified in the execution JCL on the DD statement named INPUT1.

To define the same input file in a z/VSE environment, the block size (400) and logical unit (20) must also be specified as shown below:

```
INPUT 80 400  DD=INPUT1,20
```

Example 2

```
IN  324  V  328  PS(3350)  DD=SYS015,231  LT=N,22
```

In a z/VSE environment, input is an unlabeled sequential file that is identified on DD statement SYS015. This file contains variable-length records; the size of the longest record is 324 bytes, including the 4-byte record descriptor word. The longest physical block is 328 bytes; this value includes the block control prefix. The file is stored on logical unit SYS231, a 3350 disk volume. The first 22 tape marks are bypassed as specified by the LT= keyword expression.

Example 3

```
INPUT  500  5000  PS  DD2=SYS016,16
```

Input is obtained from two sequential files in the z/VSE environment. CA Culprit reads file SYS010 on default logical unit SYS010 and then reads file SYS016 on logical unit SYS016. CA Culprit continues to alternate between these files until the operator requests end-of-file conditions by responding with an N to a CA Culprit prompt. With disk files, the first file is read from SYS010, the second from SYS016, the third from SYS017, and so forth. At the end of each file, CA Culprit prompts the operator to specify another read operation, as appropriate.

Example 4

```
 IN  120  UM(INTPROG)  PW=LETMEIN
*$THIS USER INPUT MODULE SUPPLIES ALL INPUT TO THIS RUN
```

User-written module INTPROG is a sequential input file composed of 120-byte fixed-length records; password verification is required to access INTPROG. To accommodate a comment, the user specified a continuation line by coding an asterisk (*) in column 1; a dollar sign ($) indicates a comment.

Example 5

```
INPUT FN=PAYROLL-MASTER
```

PAYROLL-MASTER is the name of a file defined to IDD. IDD automatically supplies the information CA Culprit requires to read the file (for example, record size, record type, and file type).

More information:

REC Parameter (see page 52)
CA Culprit Security Considerations (see page 192)

REC Parameter—Overview

More information:

REC Parameter (see page 212)
REC Parameter (see page 298)

REC parameters define input field characteristics, such as field length, field position, and data type. Only those input fields used during processing must appear on REC parameters; other fields in the input record can be omitted. Once defined in a CA Culprit job, an input field is available to all reports in the run.

REC parameters are automatically generated for files defined to IDD (identified by an INPUT parameter that specifies the keyword expression, FN=*file-name*). A REC parameter is generated for each field referenced by other CA Culprit parameters unless the user codes a work field or REC parameter for the field.

Input Fields

An input record may contain the following types of input fields:

- A **singly-occurring** field that occurs once in the input record.
- A **multiply-occurring** field that occurs multiple times in the input record. A multiply-occurring field can occur a **fixed** or **variable** number of times. A group of associated multiply-occurring fields is identified by the keyword GROUP on a REC parameter; a multiply-occurring field is identified by the keyword ELMNT.

Floating Groups

Any field that follows a multiply-occurring field that repeats a variable number of times is called a **floating group**; the floating group field can be a singly-occurring input field or a multiply-occurring field that repeats a fixed or variable number of times. The location of a floating group in the input buffer must be defined through the location of preceding groups.

Syntax for the REC parameter is shown on the following page.

REC Parameter

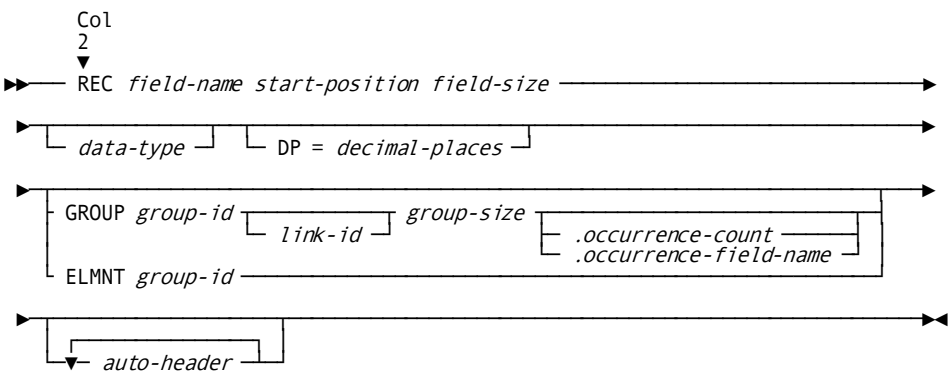
More information:

REC Parameter (see page 212)

Purpose

Defines input field characteristics.

Syntax



Syntax Rules

REC

Specifies the parameter type. It must be coded starting in column 2.

field-name

A 1- to 32-character name that identifies the input data field. It can be coded in any column after REC. The name can consist of letters, numbers, or hyphens. The following rules apply to specifying *field-name*:

- At least one alphabetic character is required.
- The name cannot begin or end with a hyphen.
- The name cannot be any of the reserved words listed in Reserved Words (see page 369).

When the GROUP keyword is specified on a REC parameter, *field-name* identifies a group of multiply-occurring input fields or a floating group. The field name cannot be referenced on any other CA Culprit parameters. When the ELMNT keyword is specified on a REC parameter, *field-name* identifies a field within a group.

start-position

Identifies the location of the input field relative to the beginning of the input record. Depending on the record type, the start position of the first input field is specified as follows:

- For **undefined (U)** or **fixed-length (F)** record types, the first field of the input record begins in position 1.
- For **variable-length (V)** record types, the first field of the input record begins in position 5; the first four bytes contain the record descriptor word (RDW).

One field can overlap another field. An overlapping input field implicitly redefines the field. With overlapping input fields, a single byte position in a record is defined with different names and characteristics. Users can redefine a single input field any number of times.

When the GROUP keyword is specified on the REC parameter, *start-position* identifies the location of the group relative to either the beginning of the record or the previously defined group. The start position is the first byte of the group relative to the beginning of the record, unless a variably-repeating group precedes this group in the record, in which case *start-position* is always 1. This value represents a start position relative to the end of the immediately preceding group.

When the ELMNT keyword is specified on a REC parameter, *start-position* identifies the location of the field, relative to the beginning of the group. The first field within a group always has a start position of 1.

field-size

Specifies the length of the input field in bytes. *Field-size* is omitted on REC parameters that specify the GROUP keyword. The following table shows the range of sizes valid for *field-size* by data type.

Code	Data Type	Size in Bytes	Sign	Input File Representation
Omitted	Alphanumeric	1-32,760		1 character per byte
1	Binary	1-8	Signed if an even number of bytes	8 binary positions per byte
2	Zoned decimal (EBCDIC)	1-31	Yes	1 digit per byte
3	Packed decimal	1-16	Yes	2 digits per byte; the low-order portion of the last byte contains a A, B, C, D, E, or F
4	Unsigned packed decimal	1-15	No	2 digits per byte
5	Multi-bit binary	1-32 bits	No	Bit string up to 32 bits

data-type

Identifies the way numeric data is stored within a field:

Code	Data type
1	A binary input field
2	A zoned decimal input field
3	A packed decimal input field

Code	Data type
4	An unsigned packed decimal field
5	A multi-bit binary field

The field size for a multi-bit binary field is 2 to 4 digits:

- **For a 2-digit field**, the first digit indicates the starting bit (1 through 9), left to right, and the second digit indicates the bit length (maximum of 9) of the field.
- **For a 4-digit field**, the first two digits indicate the starting bit (1 through 32), left to right, and the last two digits indicate the bit length (maximum of 32) of the field.

Refer to the table under **field-size** for more information on specifying data types. *Data-type* is not coded for alphanumeric fields or for fields defined on a REC parameter that specifies the GROUP keyword.

CA Culprit does not support floating point input fields. However, the CA-supplied procedure module, CULLUS36 (floating point conversion), converts single- or double-precision floating point fields to 16-byte signed packed decimal numbers with 18 significant digits. CA-supplied procedure modules can also convert packed decimal input fields to binary or zoned decimal formats; these procedure modules are CULLUS33 (numeric field conversion) and CULLUS34 (zoned decimal formatting), respectively.

DP = ***decimal-place***

Indicates the number of digits to the right of an implied decimal point in a numeric input field. This specification is omitted on REC parameters that specify the GROUP keyword.

If *decimal-place* is not coded for a numeric input field, CA Culprit treats the field as an integer in all calculations used in selection or procedure logic and in automatic totaling. If specified, *decimal-place* must be a number in the range 0 through 31.

GROUP

Identifies either a block of related multiply-occurring input fields or a floating group.

group-id

Specifies a 2-byte alphanumeric field that identifies the group. *Group-id* must be unique in the CA Culprit run.

When CA Culprit automatically generates REC parameters for fields defined to IDD, it assigns group ids sequentially, starting with 00. If more than 100 groups are generated, CA Culprit assigns lowercase alphabetic group ids.

link-id

Specifies the ID of the variably-repeating group on which the starting position of the floating group is based. This specification is used only with floating groups.

group-size

Specifies the length of a single occurrence of repeating data within the group, to a maximum of 32767 bytes.

occurrence-count

Specifies a fixed number of field repetitions; it is a number in the range 1 through 32767. A period (.) must separate this value from *group-size*.

This specification can be omitted for floating groups that occur one time. If a value of 1 is specified for *occurrence-count*, references to fields within the group must include a subscript.

occurrence-field-name

Specifies the name of a numeric input field in the fixed portion of the record; the value of this field determines the number of repetitions of a variably-repeating group.

ELMNT

Identifies an elementary item within a group of multiply-occurring fields and applies only when the associated group has been defined. One ELMNT parameter is required for each field within a group.

group-id

Identifies the ID of the group associated with the field named on the REC parameter containing the ELMNT keyword.

auto-header

An alphanumeric literal, enclosed in single quotation marks, that can be used as a column header on a printed report. Auto headers are omitted on REC parameters that specify the GROUP keyword. Field definitions on generated REC parameters can have auto-headers associated with them in the data dictionary.

The following coding considerations apply:

- A maximum of eight auto-header literals can be specified for each field definition.
- A maximum of 90 characters can be specified for each field definition.
- Headers must follow all other specifications on the REC parameter except for comments.
- A group of auto-header literals for a field can extend to a continuation line, but an individual literal cannot be broken between lines. The closing quotation mark that follows a literal must not extend beyond column 72.
- To print an apostrophe (a single quotation mark) within a literal, specify two consecutive single quotation marks.

An auto-header that is defined on a REC parameter does not print unless it is referenced on a type 5 or type 6 edit parameter.

Examples

Sample REC parameters are shown and described below.

Example 1

```
REC  PURCHASE-ORDER-TOTAL-PRICE  54  4  3  DP=2
```

PURCHASE-ORDER-TOTAL-PRICE is a 4-byte packed signed decimal field with two decimal places; the field begins at byte 54 of the input record.

Example 2

```
REC  ACCTNO  1  7  'CUSTOMER''S'  'ACCOUNT NUMBER'
```

This REC parameter defines ACCTNO as a 7-character alphanumeric field that begins at the first byte of the input record. An associated auto-header that contains two literals (CUSTOMER'S and ACCOUNT NUMBER) is also defined.

If the auto-header is requested on an edit parameter, a 2-line heading is centered above the column of account numbers; CUSTOMER'S prints above ACCOUNT NUMBER.

Example 3

```
REC  END-OF-FIELD-INDICATOR   14  54   5
```

END-OF-FIELD-INDICATOR defines a field that occupies four bits (5 through 8) of the fourteenth byte of the input record. This field can be tested for numeric values in the range 0 through 15 or for binary values in the range 0000 through 1111.

Example 4

```
REC BINARY-FIELD  0 1 1
```

BINARY-FIELD defines a 1-byte field that begins in position 0; the data is in binary format. This field can be used in match-file runs to test the status of each file.

Example 5

The REC parameters listed below describe an input record that contains a field that repeats a fixed number of times:

```
REC  POLICY-NUMBER      1  5  3
REC  COVERAGE-CODE      6  3
REC  EFFECTIVE-DATE     9  6  2
REC  MAX-LIABILITY     15  5  3
REC  ABSORPTION-GROUP  20              GROUP AA 4.4
REC  ANNUAL-COVERAGE    1  4  3  DP=2  ELMNT AA
REC  RISK-LEVEL        36  1
```

The COBOL record description for these REC parameters appears below:

```
01  COVERAGE-DATE
    05  POLICY-NUMBER    PIC S9(9)      COMP-3.
    05  COVERAGE-CODE    PIC XXX.
    05  EFFECTIVE-DATE   PIC 9(6).
    05  MAX-LIABILITY    PIC S9(9)      COMP-3.
    05  ANNUAL-COVERAGE  PIC S9(5)V99   COMP-3 OCCURS 4.
    05  RISK-LEVEL       PIC X.
```

ABSORPTION-GROUP is a group that contains a multiply-occurring input field that occurs four times; the length of each group occurrence is four bytes. ABSORPTION-GROUP begins in position 20 of the input record.

ANNUAL-COVERAGE is an elementary item of group AA. It begins in position 1 of group AA; the field is a 4-byte packed decimal field that specifies two decimal places.

RISK-LEVEL is a singly-occurring field that follows group AA. It is assigned a fixed start position because it follows a multiply-occurring group that occurs a fixed number of times.

Example 6

The following REC parameters are similar to those used in the previous example; in this example, the record contains a group of fields that occur a varying number of times:

```
REC POLICY-NUMBER        5  5  3
REC COVERAGE-CODE       10  3
REC EFFECTIVE-DATE      13  6
REC LOCATION-NUMBER     19  2  3
REC LOCATION-INFO       21           GROUP AA  14.LOCATION-NUMBER
REC LOCATION-ID          1  2        ELMNT AA
REC LOC-EFF-DATE         3  6  2     ELMNT AA
REC LOC-EXP-DATE         9  6  2     ELMNT AA
REC OTHER DATA           1           GROUP BB  AA 6
REC MAX-LIABILITY        1  5  3     ELMNT BB
REC RISK-LEVEL           6  1        ELMNT BB
```

The COBOL record description for the above REC parameters appears below:

```
01  COVERAGE-DATA.
    05 POLICY-NUMBER     PIC S9(9)    COMP-3.
    05 COVERAGE-CODE     PIC XXX.
    05 EFFECTIVE-DATE    PIC 9(6).
    05 LOCATION-NUMBER   PIC S9(3)    COMP-3.
    05 LOCATION-INFO     OCCURS 1 TO 25 DEPENDING ON LOCATION-NUMBER.
        10 LOCATION-ID   PIC XX.
        10 LOC-EFF-DATE  PIC 9(6).
        10 LOC-EXP-DATE  PIC 9(6).
    05 MAX-LIABLILITY    PIC S9(9)     COMP-3.
    05 RISK-LEVEL        PIC X.
```

POLICY-NUMBER begins in byte 5 of the input record because the file contains variable-length records. The record descriptor word occupies the first four bytes.

LOCATION-INFO is a variably-repeating group that begins in byte 21 of the input record; each occurrence of LOCATION-INFO is 14 bytes long. The value of LOCATION-NUMBER determines how many times LOCATION-INFO repeats.

LOCATION-INFO contains three elementary items identified by ELMNT AA. LOCATION-ID is defined as alphanumeric; the two date fields are defined as zoned decimal.

OTHER-DATA defines a floating group because it follows a variably-repeating group. OTHER-DATA is assigned a start position of 1 relative to the end of LOCATION-INFO; it is linked to LOCATION-INFO by the link-id, AA. The length of group BB is 6 bytes. Because group BB occurs once, the number of group repetitions is omitted. If 1 was specified for the number of group repetitions, every reference to MAX-LIABILITY and RISK-LEVEL would have to be subscripted.

SELECT / BYPASS—Overview

More information:

SELECT / BYPASS Parameters (see page 263)

What They Do

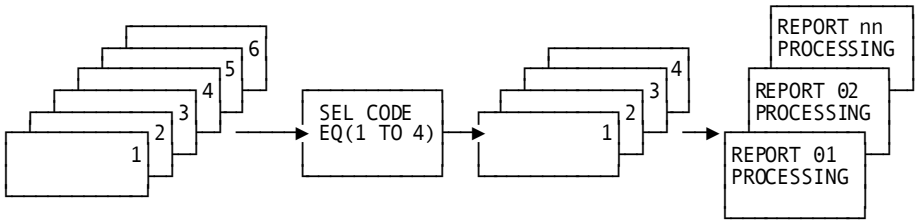
SELECT/BYPASS (SEL/BYP) parameters establish selection criteria that are applied to input records at run time. When SELECT/BYPASS parameters are included, users can select or deselect records required for a particular CA Culprit run. SELECT/BYPASS parameters apply to all reports processed for a CA Culprit run; selection logic need not be repeated during record processing for multiple reports.

SELECT and BYPASS parameters process input records:

- **SELECT (SEL)** selects and retains all records that pass specified test criteria for report processing.
- **BYPASS (BYP)** bypasses and eliminates all records that pass specified test criteria from report processing.

When to Use Them

Preference for SELECT or BYPASS depends on which method more clearly states the test conditions. Either parameter type can be coded more than once for the same input file; however, both SELECT and BYPASS parameters must not be coded for the same input file. The following figure shows how selection criteria apply to an input file.



Note: Input records that have CODE equal to a value in the range 1 through 4 inclusive are selected for processing. The selected records are passed to each report in the Culprit run.

Syntax for the SELECT/BYPASS parameter follows.

SELECT / BYPASS Parameters

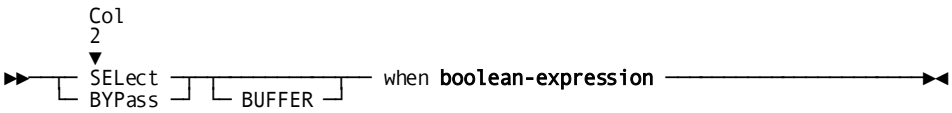
More information:

SELECT / BYPASS Parameters (see page 263)

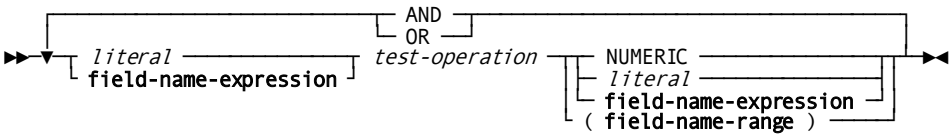
Purpose

SEL/BYP parameters establish selection criteria that are applied to input records at run time.

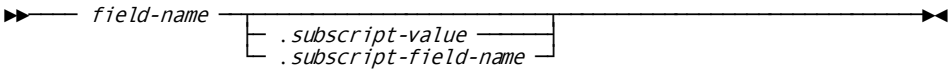
Syntax



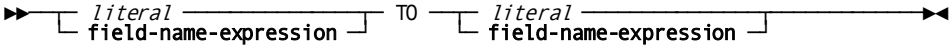
Expansion of Boolean-expression



Expansion of Field-name-expression



Expansion of Field-name-range



Syntax Rules

SELect/BYPass

Identifies the parameter type. One of these keywords must be coded starting in column 2.

BUFFER

Directs CA Culprit to apply selection criteria defined on this parameter on the completed input buffer. This option is generally used after the matching logic of a match-file run resolves the contents of the input buffer.

WHEN

An optional keyword placed before the test conditions.

boolean-expression

Represents the expanded syntax shown below. CA Culprit tests this expression at run time when it retrieves a record or fills an input buffer. If the tested expression is true, CA Culprit selects or bypasses the record or buffer, as determined by the parameter type (SELECT or BYPASS).

literal

Specifies the value of the left operand to be compared:

Literal	Meaning
alphanumeric	A 1- to 64-character alphanumeric value, enclosed in single quotation marks, that consists of letters, numbers, and special characters in any combination. To express an apostrophe (a single quotation mark), code two consecutive single quotation marks; for example, 'ERNIE''S DINER'.

Literal	Meaning
numeric	A 1- to 31-digit numeric value, optionally preceded by a sign and optionally containing an embedded or trailing decimal point; for example, 12.34.
hexadecimal	A 1- to 50-character (25 byte) hexadecimal value, preceded by X and enclosed in single quotation marks; for example, X'0000'.

field-name-expression

Specifies a field name value in the left operand that is to be compared. See expanded syntax for **field-name-expression** below.

field-name-range

Identifies a range of values to be compared. See the syntax diagram above.

The value of the left operand can be tested for an EQ or NE condition against the range of values.

Expansion of Field-name-expression

field-name

Identifies the name of either a singly- or multiply-occurring input field defined on a REC parameter or a global work field defined on a work field parameter.

subscript-value

A numeric literal that identifies a specific occurrence; for example, ACCOUNT-NUMBER.4 identifies the fourth occurrence of ACCOUNT-NUMBER.

subscript-field-name

The name of a singly-occurring input field or global work field whose value identifies the specific occurrence; for example, if INDX is assigned a value of 2, then ACCOUNT-NUMBER.INDX identifies the second occurrence of ACCOUNT-NUMBER. The definition of a field that acts as a subscript must not specify a decimal point.

A subscript value must be separated from the name of the multiply-occurring field by a period (.). Subscript values should not exceed the number of field repetitions. For example, if ACCOUNT-NUMBER occurs ten times, the subscript value should be a number in the range 1 through 10. The value of the subscript can be tested in procedure logic.

test-operation

The comparison operator for *boolean expression*. It indicates the type of test to be performed:

Symbol	Synonym	What it means
EQ	E or=	Indicates the values of the left and right operands are equal.

Symbol	Synonym	What it means
NE	N or #	Indicates the values of the left and right operands are not equal.
GT	H or >	Indicates the value of the left operand is greater than the value of the right operand.
LT	L or <	Indicates the value of the left operand is less than the value of the right operand.
GE	>= or =>	Indicates the value of the left operand is greater than or equal to the value of the right operand.
LE	<= or =<	Indicates the value of the left operand is less than or equal to the value of the right operand.

A maximum of 32 test conditions can be defined on a single SELECT/BYPASS parameter. One or more test conditions can be enclosed in parentheses. CA Culprit evaluates expressions within parentheses before evaluating expressions not enclosed in parentheses.

NUMERIC

A keyword value for the right operand. NUMERIC indicates that the value of the left operand is to be tested for numeric data. Valid test operations with this keyword are EQ and NE. An EQ test is true if the value of the left operand is numeric; a NE test is true if the value of the left operand is not numeric. If nonnumeric data is discovered, the record that contains the nonnumeric data can be dropped from further processing to avoid a data exception.

Data defined as zoned decimal is treated as numeric unless one of the following conditions applies:

- The zone portion of the last byte (that is, the high-order four bits) does not contain a valid numeric sign.
- The digit portion of any byte (that is, the low-order four bits) does not contain a value in the range 0 through 9.

The conditions stated above imply that zoned decimal fields that contain exclusively alphabetic data are interpreted as numeric. For example, the character string ABC is represented in hexadecimal as C1C2C3, which is the equivalent of +123.

literal/field-name-expression

Identifies a single value or a list of values to be compared in the right operand.

A list of values must be enclosed in parentheses; a comma or space must separate each listed value. An EQ test condition implies an OR connector between the list of values; a NE test condition implies an AND connector between the list of values.

AND/OR

Logically connects complex selection criteria. With AND, both conditions must be true; with OR, one condition must be true.

CA Culprit breaks down complex selection criteria into simple statements; these statements appear on the Input Parameter Listing as shown in the screen capture which follows below. In compound expressions, CA Culprit evaluates tests joined by AND before tests joined by OR. Multiple AND or OR connectors are evaluated from left to right.

If more than one SELECT (or BYPASS) parameter is defined for the same file, CA Culprit evaluates these statements as though they were connected by OR. In boolean expressions that contain a list of field names or literals, an EQ test condition implies an OR connector between the list of values; a NE test condition implies an AND connector between the list of values.

Input Parameter Listing of a SELECT Operation:

```
mm/dd/yy                    INPUT PARAMETER LISTING           Vnn.n  PAGE    2
********
SEL/BYP  REF       CONDITION
********
 SEL            $       SALARY EQ (35000 TO 55000) ──┬── User input
 SEL            $          OR SALARY LE 15000      ──┘
 SEL    00001      SALARY    GE        35000  ──┐
 SEL    00002      SALARY    LE        55000    ├── CA-Culprit's interpretation
 SEL    00003      SALARY    LE        15000  ──┘   of SEL criteria
```

Note: CA Culprit resolves complex selection criteria into simple statements.

Examples

Sample SELECT/BYPASS parameters are described below.

Example 1

```
SELECT (COMPANY EQ 'A' OR BRANCH EQ 35) AND STATUS NE 'L'
```

CA Culprit selects for report processing only those records for company A or for branch 35 whose status is not L.

An alternative method for selecting the same records for report processing appears below:

```
BYPASS (COMPANY NE 'A' AND BRANCH NE 35) OR STATUS EQ 'L'
```

Example 2

```
SELECT  OUT-BAL GE 1000.00
SELECT  AGE GT 60 AND OUT-BAL GT 500.00
```

Records with an outstanding balance greater than or equal to $1000.00 or with accounts older than 60 days and outstanding balances greater than $500.00 are selected.

Example 3

```
BYPASS CUSTOMER-ACCOUNT-NUMBER  EQ  (500000 TO 599999)
```

All records with account numbers in the range 500000 through 599999 are eliminated from further processing.

Example 4

```
SELECT  BUFFER  WHEN  FILE1-DATE EQ FILE2-DATE
```

After matching logic constructs the input buffer, records are selected when the contents of the FILE1-DATE field are equal to the contents of the FILE2-DATE field.

Examples of Coding Input Definition Parameters

Sample CA Culprit code containing INPUT, REC, and SELECT/BYPASS input definition parameters are shown and described below.

Example 1

```
IN 120  F  VS(KS)
REC EMP-NAME      1  25           'EMPLOYEE' 'NAME'
REC EMP-SALARY   31   5  3 DP=2   'ANNUAL' 'SALARY'
REC START-DATE   36   6           'EMPLOYEE' 'START DATE'
REC START-MONTH  36   2  2
REC START-DAY    38   2  2
REC START-YEAR   40   2  2
SEL START-YEAR EQ (79 TO 84) AND EMP-SALARY GT 50000
```

Input is contained in a key-sequenced VSAM file that contains fixed-length records; each record in the file is 120 bytes. Because the file is in the z/OS environment, *block-size* is not specified.

REC parameters defining the fields of the input record follow the INPUT parameter. START-DATE is defined as an alphanumeric input field; however, three fields overlap START-DATE and redefine the data as zoned decimal for use in arithmetic operations and comparisons. For example, the SELECT parameter contains a numeric comparison for the values of START-YEAR. Input records are selected for personnel who were hired between 1979 and 1984 and who earn a salary greater than $50,000 per year.

Auto headers are defined for EMP-NAME, EMP-SALARY, and START-DATE. These are printed on output if referenced on a type 5 or type 6 edit parameter.

Example 2

```
IN 500 F  MK=M-ACCOUNT  MB=KEEP     $MASTER-FILE
REC M-ACCOUNT       1   7       'ACCOUNT NUMBER'
REC M-CUST-NAME     8  25       'CUSTOMER' 'NAME'
REC M-CUST-ADDRESS 33  45       'CUSTOMER' 'ADDRESS'
IN  80   F  MK=T-ACCOUNT            $TRANSACTION FILE
REC T-ACCOUNT        1   7      'ACCOUNT NUMBER'
REC T-TRANACT-CODE   8   3      'TRANSACTION CODE'
REC T-TRANACT-AMT   11   5   3  'TRANSACTION AMOUNT'
SELECT T-TRANACT-CODE EQ 'DEB'
BYPASS BUFFER WHEN M-ACCOUNT GT '50000'
```

Two input files are matched in this CA Culprit run. The master file is a sequential file that contains fixed-length records 500 bytes long; the transaction file is a sequential file that contains records 80 bytes long. The files are matched on account number as specified by the match key (MK) keywords M-ACCOUNT and T-ACCOUNT.

MB=KEEP is specified for the master file; CA Culprit keeps a record from the master file in the input buffer until M-ACCOUNT in a following master file record is either less than or equal to T-ACCOUNT in the transaction file.

Before the files are matched, records in the transaction file that specify transaction code DEB are selected. After matching logic constructs the input buffer, CA Culprit eliminates all buffers that have the an account number greater than 50000.

Note: For more information on match-file runs, see the *CA Culprit for CA IDMS User Guide*.

Example 3

```
IN 80 V DD=INPUT1
REC EMP-NAME         5 25           'EMPLOYEE NAME'
REC INSURANCE-COV    30             GROUP AA    4.5
REC ANNUAL-COV       1  4  3  DP=2  ELMNT AA
REC LOCATION-NUM     50 2  3
REC LOCATION-INFO    52             GROUP BB   8.LOCATION-NUM
REC LOCATION-ID       1 2           ELMNT BB
REC EXPIRATION-DATE   3 6           ELMNT BB
IN  2  F  2  MB=DUMMY
REC BINARY-RESULT     1 2
```

Input is a sequential file composed of variable-length records; the file is described on the DD statement INPUT1. The second INPUT parameter establishes two more bytes. The REC parameter BINARY-RESULT defines the content of the two bytes. The CA-supplied procedure module CULLUS33 (numeric field conversion) can be used to put a value in this field.

The records in the input file are variable in length because the multiply-occurring field LOCATION-INFO repeats any number of times depending on the value of LOCATION-NUM. Since the records are variable in length, EMP-NAME is defined with a start position of 5; the RDW occupies the first four bytes of each record.

More information:

Output Definition Parameters (see page 69)

Chapter 4: Output Definition Parameters

This section contains the following topics:

Overview

Output definition parameters describe physical characteristics of output reports or files and specify individual report formats. The four types of output definition parameters are listed below:

- **OUTPUT (OUT) parameters** describe the physical characteristics of output reports or files and determine whether output will contain only detail or total lines, or both. This parameter is not necessary if system defaults are appropriate for the report.
- **SORT parameters** specify the order to sort extract data; they also specify control breaks using control break codes. These codes force a control break when the value of the sort field changes. In most cases, subtotals print when a control break executes.
- **Title parameters** can be used to define a title for each report; if this parameter is specified, CA Culprit prints the title at the top of each report page, along with the report number, system date, and report page number.
- **Edit parameters** define the contents and formats of output lines; there are three types of edit parameters, as follows:
  - Type 4 edit parameters define header lines for the report.
  - Type 5 edit parameters define detail lines for the report. At least one type 5 edit parameter is required for each CA Culprit report.
  - Type 6 edit parameters define total lines for the report.

An OUTPUT parameter is required for a specific report only if the values coded on the PROFILE parameter or at installation time are inappropriate for the report.

Each type of Output Definition Parameter is described separately in this chapter. Examples of all four types are provided at the end of this chapter.

More information:

PROFILE Parameter (see page 29)

OUTPUT Parameter

More information:

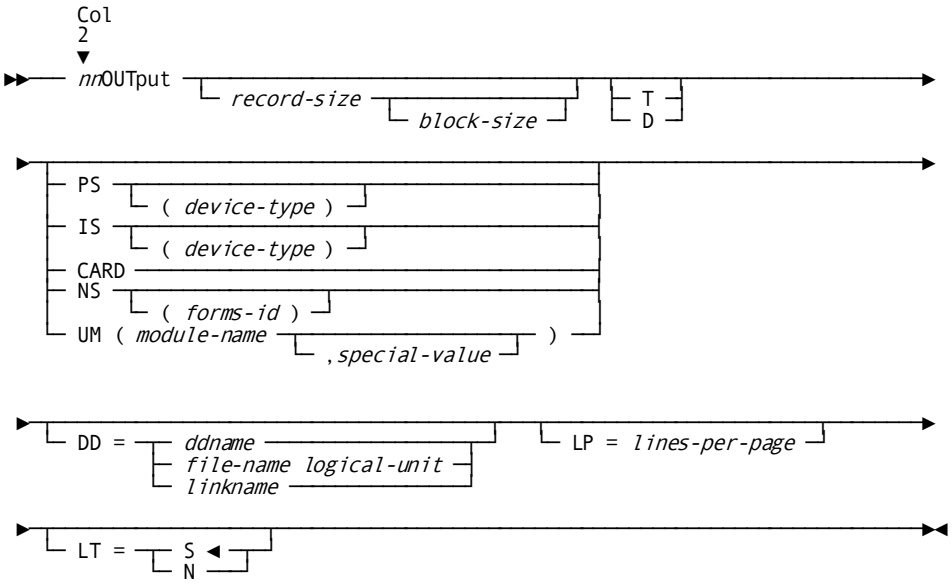
OUTPUT Parameter (see page 270)

OUTPUT Parameter (see page 300)

Purpose

OUTPUT parameters describe the physical characteristics of output reports or files, such as the number of characters per line, number of lines per page, and output file type. Additionally, the user can specify either a details-only or totals-only report.

Syntax



Syntax Rules

nn

Identifies the report associated with the OUTPUT parameter.

It is a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

OUTput

Specifies the parameter type. It must be coded starting in column 4.

record-size

Specifies the output record size for the report. It is coded in any column after OUT or OUTPUT. The default value for *record-size* is 132 bytes or the value specified on the PROFILE parameter for the run.

For printed and card output, CA Culprit adds an extra byte to the value of *record-size* to accommodate a carriage control character or stacker select character, respectively. The extra byte precedes the data in the output record.

Depending on the operating system and file type (see below), the following considerations apply:

- The maximum record size is 32,760 bytes.
- For CARD output, record-size is 80 bytes.
- For tape or disk output, record-size is the size of the longest record to a maximum of 32,760 bytes. For variable-length output, the record size must include the 4-byte record descriptor word (RDW).
- Make sure that the record length and block size of the SYS006 and SYS008 files can accommodate a record as large as the output record.

block-size

Specifies the size of a physical block of output records. In a z/VSE environment, this parameter must be specified for file types PS and IS (see below); under z/OS and z/VM block size serves for documentation purposes only.

The maximum block size is 32,760 bytes. Additionally, *block-size* must not exceed the track capacity of the applicable disk device.

T/D

Specifies a totals-only (T) report or a details-only (D) report. If this parameter is not specified, CA Culprit prints both total and detail lines.

PS/IS/CARD/NS/UM

Specifies the file type or organization of the output file; if this parameter is not specified, the output is a printed report.

Valid specifications appear below:

PS (*device-type*)

Specifies sequential file output. *Device-type* is an optional qualifier that must be enclosed in parentheses. Possible values for *device-type* are TAPE, 2311, 2314, 3310, 3330, 3340, 3350, 3370, 3375, 3380, or FBA.

Note: Under z/OS and z/VM, users can specify *device-type* for documentation purposes. Under z/VSE, CA Culprit determines the device type if the file is assigned to a particular device in execution JCL. CA Culprit requires *device-type* to be specified in either execution JCL or on the OUTPUT parameter; this rule applies even with a tape or disk management system (for example, TFAST, DFAST) that may ignore the assignment.

z/OS, z/VSE, and z/VM users can create variable-length output by using a PS output file type. Execution JCL for the output file must specify RECFM=V OR VB. The output lines contain the record descriptor word (RDW) in the first four bytes of the file; bytes 1-2 hold the record length and bytes 3-4 hold binary zeros. If the output records are the same as the input records, CA Culprit rewrites the input RDW as the output RDW. If a new RDW is required for output, CULLUS33 (numeric field conversion), a CA-supplied procedure module, can be used to convert the calculated output field length into a binary number at run time.

Note: For more information on using this module, see the *CA Culprit for CA IDMS User Modules Guide*.

IS (*device-type*)

Specifies indexed sequential (ISAM) file output. *Device-type* is an optional qualifier that must be enclosed in parentheses. Possible values for *device-type* are 2311, 2314, 3310, 3330, 3340, 3350, 3370, 3375, 3380, or FBA.

When IS is specified, a SORT parameter is required, and the primary sort field must appear on a type 5 edit parameter (see SORT Parameter—Overview (see page 76) and Edit Parameters—Overview (see page 82)). A details-only report (D) should be specified on the OUTPUT parameter to prevent automatic totaling; otherwise, a CA Culprit ISAM error may occur.

Note: Under z/OS and z/VM users can specify *device-type* for documentation purposes. Under z/VSE, CA Culprit determines the device type if the file is assigned to a particular device in execution JCL. CA Culprit requires *device-type* to be specified in either execution JCL or on the OUTPUT parameter; this rule applies even with a tape or disk management system (for example, TFAST, DFAST) that may ignore the assignment.

CARD

Specifies an output card file. If CARD is specified, record size and block size specifications can be omitted.

Note: (z/OS and z/VM only) Users must specify RECFM=FA or FBA and LRECL=81 in execution JCL for SYSPCH because the output record for CARD output includes a stacker-select character. Alternatively, a PS output file type can be specified; CA Culprit will direct output to the default or specified ddname. Execution JCL (see page 315) provides a list of default ddnames for output files.

NS (*forms-id*)

Specifies output on special forms. *Forms-id* specifies the type of special form. It is a 1- to 10-character identifier that must be enclosed in parentheses. Identifiers that contain special characters (that is, anything other than letters, numbers, or hyphens) must be enclosed in single quotation marks within parentheses.

The effect of *forms-id* depends on the SF= keyword coded on the PROFILE parameter.

Operating system considerations appear below:

- In z/OS and z/VSE environments, special forms are controlled through execution JCL; *forms-id* can be specified for documentation purposes. Under z/VSE, forms can also be controlled through CULLPOWR (VSE/POWER segmentation), a CA-supplied output module.

  Note: For details on using this module, see the *CA Culprit for CA IDMS User Modules Guide*.

- In a non-VS z/VSE environment, SF=EXTRA can be specified on the PROFILE parameter; this specification causes *forms-id* to appear on the console to prompt the operator to mount the appropriate form. Additionally, the first page of special-forms output is reprinted to allow the forms to be properly aligned; the operator can request reprints of the first page as needed.
- In a z/VM environment, special forms are controlled through the SPOOL command.

UM (*module-name,special-value*)

Specifies that edited output lines are passed to an output module for special handling. Operating system considerations appear below:

- Under z/VSE, *module-name* must be the name specified on the linkage editor PHASE statement when the module was linked.
- Under z/OS, *module-name* must be the name or alias of the load module.
- Under z/VM, *module-name* must be the name of the load module.

special-value

Specifies a 2-byte value optionally passed to an output module. The value of *special-value* and the type of data contained in this specification vary, depending upon the needs of the output module.

Note: For more information on coding special values with CA-supplied output modules, see the *CA Culprit for CA IDMS User Modules Guide*.

DD =

Specifies the name of the output file.

Execution JCL **(see page 315)** contains a list of default ddnames and filenames. When the DD= keyword expression is specified for a tape or disk report in a CA Culprit run, it must be specified on the OUTPUT parameter for all tape or disk reports in the run.

DD = *ddname*

Specifies the name of the DD statement in z/OS execution JCL or FILEDEF statement in z/VM JCL that describes the output file. This keyword expression is not valid if the file type is omitted (that is, output is a printed report).

DD = *file-name logical-unit*

Specifies the file name of the output file in a z/VSE environment and the logical unit number of the device that receives the output file.

If DD = *file-name* is specified, *logical-unit* must also be specified. *Logical-unit* is a number in the range 0 through 255.

In a z/VSE environment, a DD= specification is invalid when either the file type is omitted (that is, output is a printed report) or the file type is CARD.

LP = *lines-per-page*

Specifies the maximum number of printed lines for each report page. This specification applies only to printed reports and output modules as follows:

- For printed output, the default value for *lines-per-page* is 55 or the value specified of the PROFILE parameter.
- For output to an output module, a default value for *lines-per-page* does not exist. The LP= keyword expression must be specified in order to write title and header lines at regular intervals.

The following considerations apply to *lines-per-page*:

- CA Culprit counts detail and total lines, and their associated spacing requirements.
- CA Culprit does not count header or title lines, or their associated spacing requirements.
- Special channel skips count for one line each.

When the specified number of lines is reached, CA Culprit prints title and header lines (if any) at the top of a new page, followed by detail and total lines. The line count is reset to zero.

LT = S/N

Specifies the label type for tape file output as follows:

- S (default) indicates that standard labels are to be created.
- N indicates that no labels are to be created.

Examples

Sample OUTPUT parameters are shown and described below.

Example 1

```
25OUT  120  LP=40
```

Report 25 is a printed report that contains 120 characters on each line. Each page of the report contains up to 40 lines classified as detail lines, total lines, and break lines associated with these edit lines; additionally, header lines and a title line are printed if they were specified for Report 25.

Example 2

```
99OUTPUT  T  CARD
```

Report 99 is output to a card file. The report contains only total lines.

Example 3

```
02OUT  140  700  D  PS(TAPE)  DD=SYS036,36  LT=N
```

In a z/VSE environment, a sequential output file is written to an unlabeled tape; the file, described on DD statement SYS036, is assigned to logical unit SYS036. The file contains 140-byte fixed-length records; each block contains 5 records. The records contain details-only information.

Example 4

```
01OUTPUT  70  NS(INVOICE)  LP=28  DD=SYS018
```

Report 01 is printed on special forms. Each page of Report 01 contains 28 lines of 70 characters. CA Culprit writes this report to the file described on DD statement SYS018. DD= is a valid specification for special forms output, although it is not valid for standard printed output.

Under z/OS and z/VM, the special forms specification serves for documentation purposes only, because execution JCL handles forms control. CA Culprit does not reprint the first page and does not prompt the operator in a z/OS environment.

Example 5

```
44OUTPUT  D  UM(DTAILREP)  LP=55
```

Detail lines for Report 44 pass to a user-written output module, DTAILREP. CA Culprit inserts a page eject after every 55 lines.

More information:

Execution JCL (see page 315)
Edit Parameters — Overview (see page 82)
PROFILE Parameter (see page 29)
Restart Capability (see page 363)

SORT Parameter— Overview

SORT parameters specify a sequence for extracted data and establish control breaks when sort-key fields change. As many as 20 unique sort-keys fields can be specified for each report; each sort-key field has its own ascending or descending sequence indicator.

A control break code can be specified on each sort-key field. This code calls for a control break when the value of the sort-key field changes. In most cases, subtotals print when a control break executes. The value of the control break code determines report spacing associated with total lines.

The number of control break codes determines the maximum value of **LEVL** for the report. LEVL, a CA Culprit reserved word, is used in type 8 procedure logic.

The SORT parameter can be omitted if sorting is not required and control breaks are not needed.

Syntax for the SORT parameter is shown on the following page.

More information:

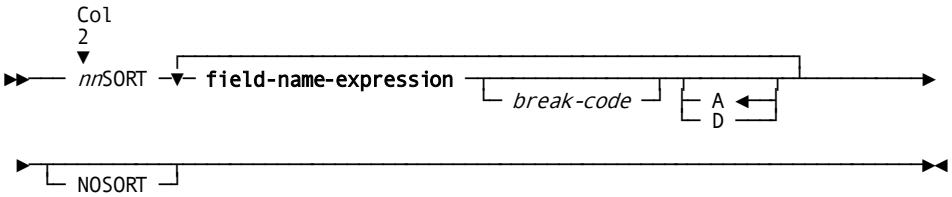
Process Parameters (see page 103)

SORT Parameter

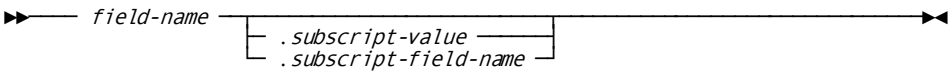
Purpose

Specifies a sequence for extracted data and establishes control breaks when sort-key fields change.

Syntax



Expansion of Field-name-expression



Syntax Rules

nn

Identifies the report associated with the SORT parameter.

Nn must be a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

SORT

Specifies the parameter type. It must be coded starting in column 4.

field-name-expression

Specifies a field to be sorted. It can be coded in any column following SORT.

See expanded syntax for **field-name-expression** below.

field-name

Specifies the name of a singly- or multiply-occurring input field defined on a REC parameter or work field defined on a work field definition parameter.

subscript-value/subscript-field-name

Specifies a subscript value for a multiply-occurring field.

If a subscript value is provided, it must be separated from the name of the multiply-occurring field by a period (.). The value of the subscript should be an integer in the range 1 through *n*, where *n* is the number of repetitions of the multiply-occurring field. For example, if EMPLOYEE occurs ten times, the value of the subscript should not exceed 10. The value of the subscript can be tested in procedure logic.

subscript-value

Specifies a numeric literal that identifies a specific occurrence of a multiply-occurring field; for example, EMPLOYEE.4 identifies the fourth occurrence of EMPLOYEE.

subscript-field-name

Specifies the name of a singly-occurring numeric input or work field whose value identifies a specific occurrence of a multiply-occurring field; for example, if INDX has a value of 2, then EMPLOYEE.INDX identifies the second occurrence of EMPLOYEE. The subscript field definition must not specify a decimal point.

break-code

Specifies a control break when the value of the sort-key field changes. It is a 1-character code whose value specifies output spacing after the control break executes;

The **table** lists valid control break codes and their effect on report spacing.

When a control break is immediately followed by a higher level control break, all lower level control breaks execute, beginning with the lowest level. The control break code associated with the highest level control break determines output spacing.

Subtotals are automatically accumulated each time a control break executes. These lines are passed to the output file unless total lines are suppressed in procedure logic or by a details-only specification on the OUTPUT parameter. If total lines are suppressed, spacing indicated by the control break code is still performed.

Each control break specified on the SORT statement has a corresponding level number, referenced by the CA Culprit reserved keyword **LEVL**. LEVL is used in type 8 procedure logic to identify the control break that is currently active.

A/D

Specifies the sequence in which values are sorted. A (default) designates an ascending order (from lowest to highest); D designates a descending order (from highest to lowest).

NOSORT

Eliminates sorting of extract data, while allowing sort-key fields and associated sort breaks to be defined. Sort-key fields can be used to make detail information available to header and totals processing.

NOSORT can appear only once on a SORT parameter. Generally it is coded after the sort-key fields.

Usage

Sort Key Considerations

A maximum of 20 field names can be specified as sort keys. The sort hierarchy is established by the order of the sort keys from left to right. The leftmost key is the primary sort key; the rightmost key establishes the most minor sort sequence of extract data. Each sort key can be followed by its own control break code and A/D designation.

The total length of all sort-key values must not exceed 240 bytes. The following table lists the number of bytes allocated to each type of sort key.

For output file type IS, the primary sort key is used to load the output file. The primary sort key must also appear on a type 5 edit parameter.

The following table also lists the lengths allocated to each type of sort-key value in order to calculate the total length of all sort keys.

Field	Definition *	Length of Sort-key Value in Bytes
REC	N	Data types 1 and 5 (binary and bit fields): 5 bytes
	N	Data types 2, 3, and 4: Half the maximum number of digits (rounded down) plus 2; for example, allow 10 bytes for a field that contains 17 digits
	A	Field length plus 1
Work field	N	Field with a decimal position or initial value exceeding 15 digits: 17 bytes
	N	Field with no decimal position and an initial value less than or equal to 15 digits: 9 bytes
	A	Field length plus 1

Note:

* A = Alphanumeric
N = Numeric

Break Code Considerations

Break Code	Spacing	Comments
1	A new page is started after the control break	CA Culprit prints any title and header lines and associated spacing at the top of the new page. Do not use this break code for nonprinted output.
0	One blank line is printed after the control break	Do not use this break code for nonprinted output; it may cause blank records to appear in the output file.
-	Two blank lines follow the control break	Do not use this break code for nonprinted output; it may cause blank records to appear in the output file.
+	No blank lines follow the control break	This code is appropriate for printed and nonprinted output.

Examples

Sample SORT parameters are shown and described below.

Example 1

```
01SORT  CITY,1  ZIP  CUST-NAME
```

Customer names are sorted in alphabetical order according to zip code, zip codes in ascending order within city, and cities in alphabetical order. The break code associated with CITY specifies a new page of output when the value of CITY changes. Each page contains title and header lines (if they are specified), detail lines, and subtotal lines. The last page will also include the grand total lines for the report.

Example 2

```
10SORT  DIVISION,-  DEPT,0  EMPLOY-NO
```

The primary sort key for Report 10 is DIVISION. Within each division, extracted records are sorted by employee number within department. All fields are sorted in ascending order.

The value of LEVL associated with a control break on DEPT is 1; the value associated with a control break on DIVISION is 2; at grand totals time, the value of LEVL equals 3. These values can be used in type 8 procedure logic to test the currently active control break.

Detail lines within department are single spaced in the output file. When a control break executes on DEPT and is not immediately followed by a control break on DIVISION, one blank line follows the subtotal lines output for DEPT. When a control break executes on DEPT and is immediately followed by a control break on DIVISION, the break code associated with DIVISION outputs two blank lines following subtotals for DEPT and DIVISION.

Example 3

```
02SORT  BRNAME,0  NOSORT
```

Extracted records for Report 02 are not sorted; however, one blank line is output after each branch name. Additionally, CA Culprit maintains the current value of BRNAME for use on a type 4 header edit parameter.

More information:

Process Parameters (see page 103)
Output Phase Field References (see page 309)
Edit Parameters — Overview (see page 82)

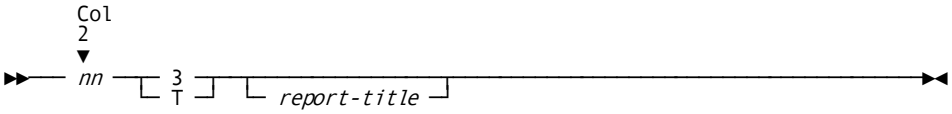
Title Parameter

Purpose

Gives a report a descriptive title. One title parameter is permitted for each report, although none is required. The title parameter is usually omitted for nonprinted output.

The title line contains the report number, the user-supplied title (if any), the system date, and the report page number. If TS= is specified on the PROFILE parameter, the system time is printed after the system date.

Syntax



Syntax Rules

nn

Identifies the report associated with the title parameter. *Nn* must be a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

3

Specifies the parameter type. It is a single character coded in column 4. Alternatively, T (title) can be used to identify a title parameter.

report-title

Specifies the user-supplied report title. It is a 1- to 50-character literal and can include embedded blanks. If specified, the title can be coded starting in any column after 4. *Report-title* is centered over the body of the report as determined by the output line size specified for the report.

The following rules apply to specifying *report-title*:

- Quotation marks are not required; if quotation marks are coded, they appear in the report title.
- Leading blanks are ignored.
- Continuation lines are not permitted.
- Comments are not permitted.
- If the title, including the report number, page number, and date stamp, exceeds the output record size, *report-title* is truncated on the right; the date and page number still print on the title line.

Example

A sample title parameter is shown below.

```
013EMPLOYEE SALARY REPORT
```

The literal EMPLOYEE SALARY REPORT is printed on the top line of each page in Report 01, as shown in the following figure. The title is centered over the body of the report; REPORT NO. 01 is printed on the same line on the left side of the page; the date in the form *mm/dd/yy* is printed on the right side, followed by PAGE and the page number of the report.

```
REPORT NO. 01                EMPLOYEE SALARY REPORT        mm/dd/yy PAGE    2
DEPARTMENT BLUE SKIES

                                                        EMPLOYEE
                        TITLE                             NAME                          SALARY

                CUMULUS CARETAKER                 BETH      CLOUD                       $52,750.00

                MGR BLUE SKIES                    DANIEL    MOON                        $72,000.00

                SUNSHINE SUPERVISOR               ALAN      DONOVAN                     $33,500.00
                                                                                       $158,250.00
```

More information:

PROFILE Parameter (see page 29)

Edit Parameters— Overview

More information:

Process Parameters (see page 103)
Edit Parameters (see page 278)
EDIT Parameter (see page 302)

Edit parameters define the output lines for a report. There are three types of edit parameters, as follows:

Header Line

A **type 4 edit parameter** defines a header output line; type 4 edit parameters are optional. Header output lines also result from auto-header specifications coded on type 5 or type 6 edit parameters.

Detail Line

A **type 5 edit parameter** defines a detail output line. At least one type 5 edit parameter is required for every report.

A type 5 edit parameter also performs automatic totaling. Automatic totaling accumulates the values of all numeric fields specified on type 5 edit parameters; the totals are output on total lines at every control break and at the end of the output phase. Automatic totaling is performed if the OUTPUT parameter does not specify a details-only report and if type 6 edit parameters are not coded for the report.

Total Line

A **type 6 edit parameter** defines a total output line; type 6 edit parameters are optional. Total lines are usually output at each control break and at the end of the output phase. Type 8 procedure logic can be used to control the output of total lines.

Each edit parameter describes the location and format of one output field:

Location

Location is determined by the following items:

- The type of edit parameter determines whether the output field appears on a header line, detail line, or total line.
- The edit parameter line number specifies one of eight possible output lines for each type of edit parameter.
- The edit parameter column number specifies the absolute or relative position the output field occupies on the output line. An absolute column number identifies the line position in which the output field begins printing. A relative column number identifies a relative column position on the output line.

Format

Format is determined by the following items:

- Field size specifies the maximum number of columns reserved for the output field on the output line.
- Decimal place specifies the number of digits that will appear to the right of the decimal point in a numeric output field.
- Specific format options specify editing formats for numeric output fields; for example, F$ edits the value 1234 as $1,234.

The syntax for edit parameters appears on the following page.

Edit Parameters

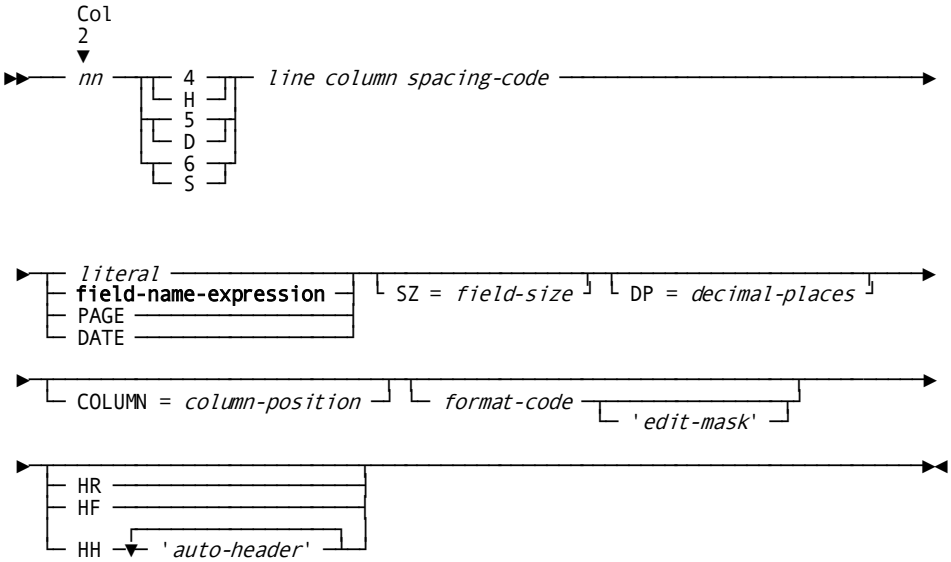
For more information, see:

Edit Parameters (see page 278)

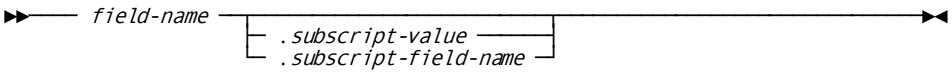
Purpose

Defines the output lines for a report.

Syntax



Expansion of Field-name-expression



Syntax Rules

nn

Identifies the report associated with the edit parameter. *Nn* must be a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

4

Indicates that the edit parameter defines an output field on a **header line**; alternatively, H (header) can be used to identify a header parameter.

Must be coded in column 4.

Report headers can be specified either on type 4 edit parameters or by using the auto-header capabilities of CA Culprit. Headers specified on type 4 edit parameters are printed above headers specified as auto-headers; auto-headers are discussed later in this chapter under syntax rules for HH, HR, and HF.

5

Indicates that the edit parameter defines an output field on a **detail line**; alternatively, D (detail) can be used to identify a detail parameter.

Must be coded in column 4.

6

Indicates that the edit parameter defines an output field on a **total line**; alternatively, S (summary) can be used to identify a total parameter.

Must be coded in column 4.

line

Specifies an edit line. It must be a single digit in the range 1 through 8 and must be coded in column 5.

Note: The total of the number of lines specified for type 4 edit parameters and the number of auto-header literals coded on a type 5 or type 6 edit parameter must not exceed 8. If the total number of equals 8, type 4 edit parameters must specify *line* sequentially, starting with 1.

column

Specifies the location of the output field on the output line. *Column* must be a 4-character value coded in columns 6 through 9. *Column* can specify either an absolute column position on the output line or a relative column position on the output line, as follows:

- An **absolute column position** identifies the output line position in which the output field begins printing. The value ranges from 0 to *n*, where *n* is the output record size specified for the report. Leading zeros can be omitted from this specification; if leading zeros are omitted, *column* must be right-justified on column 9.

  Note: To specify a column beyond 9999, see COLUMN= later is this chapter.

- A **relative column position** identifies the location of the output field on the output line relative to other output fields. Column 6 must contain an asterisk (*); columns 7 through 9 must contain a relative column number in the range 000 through 999. Leading zeros can be omitted from this specification; if leading zeros are omitted, the column number must be right-justified on column 9.

The following considerations apply to coding a relative column position:

- All fields with the same relative column number align in a single column. The column width is set to the length of the longest field or literal defined on an edit parameter coded with that particular relative column number. CA Culprit spaces these columns evenly across the output record.
- Relative column numbers do not have to be coded in a particular order. CA Culprit sorts the columns in ascending order from left to right across the output record.
- Relative column numbers do not have to be sequential; for example, a three column report can be coded as * 10, * 18, and * 21. With this feature, new columns of information can easily be added to a report.
- Relative column numbers and absolute column numbers must not be combined in the same line definition; for example, all output fields on line number 1 of a type 5 edit parameter must be coded with either a relative or an absolute column position, but not with both. This rule does not apply when *column* equals 0 (either 0000 or *000).

A column position of 0 can be specified for one or more numeric output fields coded on type 5 edit parameters. These fields are not output in the detail lines for the report. CA Culprit totals the values of the nonprinted fields; the field totals can be used during the output phase in type 8 process parameters and type 6 edit parameters.

Output fields must not overlap on the output record, with the exception of a trailing negative sign edit character. If the first character of an output field overlaps the negative sign that follows the previous output field, CA Culprit performs the following tasks:

- On a type 4 header edit parameter, the first character of the overlapping field prints, unless it is a literal. If the second field is a literal, a blank space or the trailing negative sign prints instead of the first character of the literal.
- On a type 5 or type 6 edit parameter, the first character of the overlapping field prints instead of the trailing negative sign.

spacing-code

Specifies the number of lines skipped before the edit line is output. *Spacing-code* is a single character that must be coded in column 10. Valid space codes include any standard ASA character, as listed in the following table.

If a single edit line definition contains more than one type of space code, spacing is determined by the value of the last nonblank code encountered (that is, the space code on the edit line definition that has the highest column number).

When a space code of 1 appears on a type 5 or type 6 edit parameter, the associated detail or total line prints on a new page. Title and header lines do not print, unless this feature was requested at installation.

Spacing-code Value	Spacing Performed
Omitted	Single space
0	Double space
-	Triple space
1 through 9	Skip to channel 1 through 9
A,B,C	Skip to channel 10(A), 11(B), or 12(C)
+	No spacing (overprint on the same line)
V	Select card into pocket 1 (P1); applies to card output only
W	Select card into pocket 2 (P2); applies to card output only

***literal*/*field-name-expression*/PAGE/DATE**

Specifies an output field.

literal

Specifies an alphanumeric or hexadecimal literal:

- An **alphanumeric literal** is a 1- to 64-character string enclosed in single quotation marks; to express an apostrophe (a single quotation mark), code two consecutive single quotation marks; for example, 'ERNIE''S DINER'.
- A **hexadecimal literal** is a 1- to 50-character (25-byte) hexadecimal string preceded by X and enclosed in single quotation marks; for example, X'00FF'.

field-name-expression

Specifies an input field defined on a REC parameter or a work field defined on a work field parameter.

See expansion of **field-name-expression** below.

PAGE

A reserved word that can appear only on type 4 edit or type 8 process parameters (see Error! Reference source not found.).

When PAGE is coded on a type 4 edit parameter, CA Culprit outputs a 1- to 6-digit page number with leading zeros suppressed. As each new page begins, the value of PAGE increases by 1. Users can reset this value in type 8 procedure logic, as explained in Process Parameters (see page 103).

PAGE can be defined for only one output field in a report. Generally, it is used when no title parameter is coded for a report. If PAGE is coded for a report that includes a title parameter, the page number is not output on the title line.

DATE

A reserved word that can appear only on type 4 edit parameters. When DATE is specified, an 8-character date is output in *mm/dd/yy* (American), *dd/mm/yy* (European), or *yy/mm/dd* (Canadian) format. The date format and the inclusion of the system time are governed by PROFILE parameter options DS= and TS=.

SZ = *field-size*

Specifies the number of alphanumeric characters or digits printed in the output field. If this keyword expression is not specified on a type 4 or type 5 edit parameter, the default output size is determined by the field and data type, as indicated in :tref refid=outsize..

If totals print automatically (that is, type 6 edit parameters are not coded for the report), the default output size is the size specified for the associated field on a type 5 edit parameter. If type 6 edit parameters are specified, the default output size is the size indicated in the table that follows.

The following considerations apply to *field-size* for numeric and alphanumeric fields:

- For **numeric fields**, *field-size* refers to the number of digits to be output. The output field size does not include editing characters, such as commas and decimal points. Editing characters must be considered, however, when planning output format.

  Field-size affects only the length of the field output; it does not affect the number of digits used in arithmetic calculations. When numeric fields are output, any necessary truncation or zero-filling occurs on the left.

  The following considerations apply to the SZ= keyword expression in association with specific format codes; format codes are discussed later in this chapter:

  - *Field-size* must not be specified with format codes FU, FW, FD, and FM. The size of an output field that specifies one of these format codes follows:

    ```
    FU: 4 bytes
    FW: 8 bytes
    FD: 6 bytes
    FM: user-defined
    ```

  - *Field-size* can be specified for format codes FP, FB, and FS. The following considerations apply:

    ```
    FP: Field size indicates the number of packed bytes output.
    FB: Field size indicates the number of binary bytes output.
        It must be a number in the range 1 through 4; the default
        is 4.
    FS: Field size should be specified as 9 bytes.
    ```

- For **alphanumeric fields**, *field-size* refers to the number of characters to be output. The valid range is 1- to 32,760; the default is the actual size of the field. Alphanumeric values print from left to right; they are truncated on the right if the value is longer than the size specification.

Field	Data Type	Code	Default Size
Input	Alphanumeric	Omitted	Input size
	Binary	1	Three times input size
	Zoned decimal	2	Input size
	Signed packed decimal	3	Two times input size minus 1
	Unsigned packed decimal	4	Two times input size
	Bit (data type is 5)	5	Three digits
Work	Alphanumeric		Work field size
	Numeric with no decimal point specification and containing 15 digits or less		15 digits
	Numeric with a decimal point specification or with an initial value containing more than 15 digits		31 digits

Note: The size reflects only the number of digits; CA-Culprit makes no adjustments for edit characters in output size specifications.

DP = *decimal-places*

Specifies the number of digits to the right of the decimal point in a numeric output field. *Decimal-place* must be in the range 0 to 14. If DP=0 is specified, no decimal point or decimal positions are output.

Consider the following when coding a DP= keyword expression on an edit parameter:

- If DP = *decimal-places* is specified on a REC parameter or work field parameter, the DP= specification is usually omitted from the edit parameter.
- If DP = *decimal-places* is specified on an edit parameter, the value of *decimal-places* **replaces** any explicit or implicit decimal point specification made when the field was originally defined. For example, if an input field defined with two decimal places is specified on an edit parameter with the keyword expression DP=3, the output field will have three decimal places. This replacement applies only when the field is output and not to any arithmetic operations that are performed on the field.

- If a decimal place is not specified on the parameter that defines the field or on the associated edit parameter, *decimal-places* defaults to 0 (zero).
- If *decimal-places* is greater than 14 on the field definition, the field value is output without the decimal point because the maximum number of decimal positions allowed in an output field is 14.
- DP = *decimal-places* cannot be specified on an edit parameter that specifies one of the following format codes: FU, FW, FB, FD, FM, FS, FZ, or FP. If the DP= keyword expression is specified with format code F*n*, the number of decimal places specified should be the same as or less than the value of *n*.

Sample decimal specifications appear in the following table.

Note: If the PROFILE parameter option RELEASE=5 is in effect, the default for *decimal-places* is 2 for all numeric fields unless a DP= keyword expression is specified on a REC, work field, or edit parameter for the field.

REC Parameter Specification DP=	Edit Parameter Specification DP=	Result
Not specified	Not specified	///123,456
0	Not specified	///123,456
Not specified	1	///12,345.6
3	Not specified	///123.456
3	2	//1,234.56

Note: In this table, the field size equals 8 bytes. Blanks are shown as slashes (/)

COLUMN = *column-position*

Specifies the absolute position of an output field beyond column position 9999. *Column-position* is an integer in the range 10,000 to 32,760.

The following coding considerations apply:

- When this keyword is used on type 5 and type 6 edit parameters, columns 6 through 9 on the edit parameters must be blank.
- Values for the SZ= keyword expression in the range 1 through 32,760 are valid for alphanumeric fields.
- JCL file definitions must be modified as follows:
  - Specify the size of the output record in the SYS020 file.
  - If necessary, increase the record length and block size on the SYS006 and SYS008 files to accommodate a record at least as large as the output record.

format-code edit-mask

Specifies format options for numeric output fields, as follows:

- **Format-code** is a 2-character code that identifies different format options. Possible values are listed under **Valid Format Codes for Printed Output Fields** and **Valid Format Codes for Nonprinted Output Fields**. Examples of printed output fields appear under **Examples of Format Edit Specifications for Printed Output Fields**.

  If *format-code* is not specified, leading zeros are suppressed and commas are inserted after every three digits, counting from right to left from the decimal point. If FU, FW, FB, FD, FM, FS, FZ, or FP is specified, the explicit or implicit decimal point specified when the field was originally defined is ignored. If FD, FM, or FS is specified on a type 5 edit parameter, the column total is not automatically output; the column total is output if the field is also specified on a type 6 edit parameter.

Valid Format Codes for Printed Output Fields

Code	Editing Operation Performed
Omitted	Leading zeros are suppressed; commas are inserted.
F*	F* replaces leading zeros by asterisks except for decimal positions. The last digit is followed by an asterisk, unless the field value is negative. Commas are inserted. This format option is used primarily for check protection.
F0	F0 suppresses leading zeros and omits commas. If the value is negative, a trailing minus sign is output.
F*n*	F*n* suppresses leading zeros except for the last *n* digits, where *n* is a number in the range 1 through 9. Commas are inserted. If the value is negative, a trailing minus sign is output. This option outputs a specified minimum number of zeros when the value is 0. If this format code is specified with a DP= keyword expression, the value of *n* should be equal to or greater than the number of decimal places.
F$	F$ suppresses leading zeros except for decimal positions or, if there are no decimal positions, the last digit. Commas are inserted. A floating dollar sign is inserted before the first number output. If the value is negative, a trailing minus sign is output.
FN	FN outputs leading zeros and omits commas for the default length of the field. If the value is negative, a trailing minus sign is output. A decimal point is output, if specified.
FC	FC suppresses leading zeros except for decimal positions or, if no decimal positions are specified, the last digit. Commas are inserted. If the value is negative, it is enclosed in parentheses. The left parenthesis is output immediately before the first digit output.

Code	Editing Operation Performed
F-	F- suppresses leading zeros except for decimal positions or, if no decimal positions are specified, the last digit. Commas are inserted. If the value is negative, a minus sign is output before the first digit output.
FF	FF suppresses leading zeros except for decimal positions or, if no decimal positions are specified, the last digit. Commas are omitted. If the value is negative, a minus sign is output before the first digit output.
FD	FD edits the field in date format: *mm/dd/yy*. The field size is assumed to be six digits. Keyword expressions SZ= and DP= are invalid on an edit parameter that specifies format code FD. The accumulated total of a field that specifies this format code on a type 5 edit parameter does not appear in a total line that is generated automatically.
FS	FS edits the field in social security number format: *nnn-nn-nnnn*. The field size is assumed to be nine digits. The keyword expression DP= is invalid on an edit parameter that specifies format code FS. The accumulated total of a field that specifies this format code on a type 5 edit parameter does not appear in a total line that is generated automatically.
FM	FM edits the field as specified by the associated user edit mask. A user-defined edit mask must follow. Keyword expressions SZ= and DP= are invalid on an edit parameter that specifies format code FM. The accumulated total of a field that specifies this format code on a type 5 edit parameter does not appear in a total line that is generated automatically.
FR	FR reverses the contents of an alphanumeric text string. This is useful for languages that require the reader to read right to left.

Valid Format Codes for Nonprinted Output Fields

Code	Description of Editing
FB	FB converts an integer field value to binary output. The integer field value must be a number in the range -2,147,483,648 through 2,147,483,647. The output field size is 1 to 4 bytes; 4 bytes is the default. Any necessary truncation or zero-filling occurs on the left.
FU	FU converts a numeric field value to a single precision floating point value. The output field size is 4 bytes. Keyword expressions SZ= and DP= are invalid on an edit parameter that specifies format code FU. Format code FU is used primarily in scientific applications.
FW	FW converts a numeric field value to a double precision floating point value. The output field size is 8 bytes. Keyword expressions SZ= and DP= are invalid on an edit parameter that specifies format code FW. Format code FW is used primarily in scientific applications.

Code	Description of Editing
FP	FP converts the output field value into a signed packed decimal number. This format code should be specified for nonprinted output; when used with printed output, the results are unpredictable. The maximum field size with format code FP is 16 bytes.
FZ	FZ converts the output field value into a zoned decimal number. Negative values are output with an 11-zone punch over the last digit. This format code should be specified for nonprinted output; when used with printed output, but the sign punch either suppresses printing of the last digit (if it is 0) or prints the last digit as an alphabetic character.

Examples of Format Edit Specifications for Printed Output Fields

Format Edit Code	+123456	0	-6315
F*	***123,456*	**********	*****6,315-
F0	//123456	////////	////6315-
F2	///123,456	////////00	/////6,315-
FN	00123456	00000000	00006315-
F$	//$123,456	////////$0	////$6,315-
FC	///123,456	/////////0	////(6,315)
F-	///123,456	/////////0	////-6,315
FF	//123456	///////.0	///-6315

Note: In this table, the field size equals 8 bytes. Blanks are shown as slashes (/).

edit-mask

Describes the size and format for a numeric output field. It applies only to fields defined with format code FM. *Edit-mask* is a 1- to 64-character expression, enclosed in single quotation marks, that describes the field size and format, the number of decimal places, and any characters to be inserted into the value.

Selected alphanumeric characters have unique meanings when used in an edit mask; these characters are described below:

- A **digit indicator** is represented by a **9**. A 9 is replaced, from right to left, with digits from the numeric field, including leading zeros. An edit mask can contain a maximum of 31 digit and zero-suppression indicators (see below).

- A **zero-suppression indicator** is represented by a **Z**. A Z is replaced, from right to left, with the corresponding digit, unless the digit is a leading zero. A Z replaces any zeros to the left of the first nonzero digit with blanks. An edit mask can contain a maximum of 31 digit and zero-suppression indicators.

  Note: A Z to the right of a decimal point indicator (.) or to the right of a digit indicator (9) is treated as an insertion character or negative value indicator (see below). When used in this way, Z does not function as a zero-suppression indicator.

- A **decimal point indicator** is represented by a period (.). A decimal point indicator outputs a decimal point in the position it occupies in the edit mask. All digits to the right of the decimal point are printed. The number of digit indicators to the right of the decimal point indicator should agree with the number of decimal positions specified when the field was originally defined.

- **Insertion characters** specify any characters other than digit or decimal point indicators. Insertion characters are output in the output field in the position they occupy in the edit mask. Insertion characters are not output unless a digit from the field has already been output. More specifically, these characters do not print if all preceding digits were zero-suppressed and replaced on output by blanks, as a result.

  A Z is treated as an insertion character if it occupies a position in the edit mask to the left of the rightmost digit indicator (9) and to the right of at least one digit or decimal point indicator; for example '9.Z99'. To express an apostrophe (a single quotation mark) in an edit mask, code two consecutive single quotation marks.

- **Negative value indicators** specify a negative value. These indicators can be any character (including Z) or combination of characters that follows the rightmost digit indicator. Such characters print as they appear in the edit mask only if the output field has a negative value. There is no default negative value indicator if an edit mask is used.

Each edit mask must begin with a digit indicator, a zero-suppression character, or a decimal point indicator; all other types of characters are optional. Sample edit masks are shown in the following table.

Indicator	User Mask	Field Value	Printed Result
Digit	'999' or 'ZZZ'	+123	123
	'9999'		0123
	'ZZZZ'		/123
	'99' or 'ZZ'		23
	'ZZZ99'		//123
	'9999'	-123	0123

Indicator	User Mask	Field Value	Printed Result
Decimal point	'9.99'	+123	1.23
	'Z.999'		/.123
Insertion character	'99A99B99C'	+123	00A01B23
	'99Z99'		01Z23
Negative value	'9.99-'	+123	1.23
		-123	1.23-
	'ZZ.99RZ'	+123	/1.23//
		-123	/1.23RZ

HR/HF/HH *auto-header*

Specifies the origin of automatically generated headers. If auto-headers are not specified, all headers printed are from user-coded type 4 edit parameters.

Note 1: An auto-header must not be coded on a type 6 edit parameter unless a totals-only report is defined. Headers can be defined on either a type 5 or type 6 edit parameter for a totals-only report, but not on both.

Note 2: An auto-header must not be coded on a type 4 edit parameter.

HR

Indicates that the source of auto-headers is the REC parameter that defines the associated field. If HR is not specified, any auto-header literals defined on the REC parameter are ignored.

HF

Indicates that the source of the auto-header is the name of the field or literal defined on the edit parameter. For example, if the field name is ACCTNO, this name is used as the header. If the field is subscripted, the base field name (that is, the field name without the subscript value) is used as the header. If *literal* is specified, the literal name acts as the header.

HH *auto-header*

Indicates that an auto-header immediately follows HH on a type 5 or type 6 edit parameter.

The following considerations apply to coding *auto-header*:

- Auto headers must follow all other specifications of the edit parameter, except for comments.
- A maximum of 8 auto-header alphanumeric literals can be specified on each edit parameter; each literal must be enclosed in quotation marks.
- A maximum of 90 characters can be specified for the entire group of auto-header literals.
- Each literal of an auto-header must appear entirely on one line.
- To print an apostrophe (a single quotation mark) within a literal, specify two consecutive single quotation marks; for example, 'ERNIE''S'.

Auto headers appear at the top of each report page, below the title line (if any), and below headers generated by type 4 edit parameters (if any). Each auto-header is centered over the field column. If the auto-header contains more than one literal, the literals are stacked vertically to form a multiline heading.

Auto headers should be associated with a single edit line. If this is not possible, CA Culprit generates separate header lines for each edit line number with an associated header-origin code, starting with the lowest line number.

Examples

Sample edit parameters for header lines, detail lines, and total lines are shown and described below.

Example 1: Header Lines

```
01410040-'CURRENT BALANCE'
01410062 'DATE'
```

Header line 1 contains two output fields; both are literals. CURRENT BALANCE prints beginning in column number 40 and DATE prints beginning in column 62. The line is printed at the top of each new page after the title, if one is specified. Two blank lines separate the title from the header, as established by the space code - (hyphen).

An alternative way of coding the same header line appears below:

```
01H10040-'CURRENT BALANCE       DATE'
```

Example 2: Header Lines

```
3341*010     'BRANCH'
3342*010     BRANCH
```

Report 33 contains two header lines; the literal BRANCH prints in the relative column identified by 10; the branch name prints in the same relative column, directly below the literal. The field BRANCH must be specified on either a SORT parameter or in a HEAD instruction.

Example 3: Detail Lines

```
095100120 CUST-NAME            SZ=15
09520012  CUST-AMT-SALES       F$
09520035  CUST-NUM-SALES       F1
```

The body of Report 09 contains a listing of customer names, the amount of sales, and the number of sale transactions. The customer's name prints for 15 spaces on the first detail line, starting in column number 12. A blank line is output before the name is output.

The amount of sales prints directly below the name, beginning in column 12. CUST-NUM-SALES prints on the same line as CUST-AMT-SALES, starting in column 60. The sizes of these two output fields are the default values determined by the definitions of the fields. Format codes are used to edit the printed output.

CA Culprit automatically totals and prints the totals of CUST-AMT-SALES and CUST-NUM-SALES. Printing can be suppressed either by coding type 6 edit parameters or by defining a details-only report on the OUTPUT parameter.

Example 4: Detail Lines

```
 23510000 COUNT
 2351*010 CUST-NAME        SZ=15     HR
 2351*030 CUST-AMT-SALES   FM '$ZZ,ZZ9.99CR'
*                                    HH 'AMOUNT OF' 'SALES'
 2351*020 CUST-NUM-SALES   FM 'ZZ9'  HH ' ','NUMBER OF','SALES'
 2351*040 SALE-DATE        FD        HF
```

Four output fields are printed in relative column positions 010 through 040 on one detail line. The four columns are spaced evenly across the page in ascending order.

A column heading is specified for each output field. The auto-header for CUST-AMT-SALES, coded on a continuation line, contains two literals; therefore, a two-line header will appear over relative column 2. A blank line automatically prints between the column headers and first detail line because fewer than eight auto-header lines are specified.

Edit masks are used in this example to format the printed output fields. Although CUST-AMT-SALES, CUST-NUM-SALES, and SALE-DATE are numeric fields, the column totals of these fields are not automatically printed because format codes FD and FM are specified. The output field COUNT is not printed on a detail line because the type 5 edit parameter specifies 0 for a column number. However, the accumulated value of COUNT is available for use during the output phase.

Example 5: Total Lines

```
2361*020  CUST-NUM-SALES    FM 'ZZZ9'
2361*030  CUST-AMT-SALES    FM '$ZZZ,ZZZ,ZZ9.99CR'
23620001  'TOTAL CUSTOMER VISITS'
23620023  COUNT  SZ=6  F1
```

These edit parameters define total lines for the previous example. Total lines are not automatically output because type 6 edit parameters are defined. Instead, the output of total lines can be controlled with type 8 procedure logic.

Edit masks are specified again for CUST-NUM-SALES and CUST-AMT-SALES. The size of the output fields is larger to accommodate total values.

Total line 1 is coded with relative column numbers and total line 2 with absolute column numbers. A literal prints starting in column 1 of the second total line. The total value of COUNT, accumulated as a nonprinted field on a type 5 edit parameter, prints starting in column 23 of the second total line.

Coding Examples of Output Definition Parameters

Samples of CA Culprit code using OUTPUT, SORT, title, and edit parameters are shown and described in the following pages.

Example 1

```
01OUT 80 400  D  PS(3375) DD=SYS021,21
01SORT  DEPARTMENT SALARY
0151*010 DEPARTMENT    SZ=20
0151*020 EMP-NAME      SZ=25
0151*030 SALARY        SZ=10 FP
```

Output records are directed to a sequential file in a z/VSE environment. The records are 80 bytes long with 5 records to a block. SYS021 is stored on logical unit 21, a 3375 disk volume.

Control break codes are not specified on the SORT parameter for the following reasons:

- Report 01 is a details-only report; a control break code is usually specified to generate total lines.
- Report 01 is a nonprinted report; a control break code, other than a plus sign (+) would generate blank records after every control break.

A title and header lines for Report 01 are also omitted because the output records are directed to a sequential file; title and header lines are usually specified for printed reports.

Each detail line for Report 01 contains three columns of information. The first two output fields contain alphanumeric data; the output field for SALARY contains a signed packed decimal number.

Example 2

```
01SORT DEPARTMENT,-,SALARY,D
013EMPLOYEE SALARIES BY DEPARTMENT
0151*010 DEPARTMENT    SZ=20      HR
0151*020 EMP-NAME      SZ=25      HH 'EMPLOYEE NAME'
0151*030 SALARY     F$ SZ=10 DP=2 HF
```

This example is similar to the previous example, except that the output is printed. Since an OUTPUT parameter is not included, the output is a printed report with 132 characters per line and 55 lines per page. The detail lines for Report 01 sort by salary in descending order within each department. Each time the department changes, the total salary for the department prints followed by two blank lines; the control break is forced by the control break code associated with DEPARTMENT.

The following figure shows a page of output for Report 01. It contains a title line, header lines (specified by the header origin codes on the type 5 edit parameters), and total lines (automatically generated for the SALARY field). SALARY is edited according to the edit options specified on the type 5 edit parameter.

```
REPORT NO. 01        EMPLOYEE SALARIES BY DEPARTMENT                mm/dd/yy PAGE    2
                     DEPARTMENT                        EMPLOYEE NAME                    SALARY

                INTERNAL SOFTWARE                 PERCY     GRANGER                    $34,500.00
                INTERNAL SOFTWARE                 JANE      DOUGH                      $33,000.00
                INTERNAL SOFTWARE                 JAMES     GALLWAY                    $33,000.00
                INTERNAL SOFTWARE                 VLADIMIR  HEAROWITZ                 $33,000.00
                INTERNAL SOFTWARE                 RALPH     TYRO                       $20,000.00

                                                                                       $390,500.00


                PERSONNEL                         ELEANOR   PEOPLES                    $80,000.00
                PERSONNEL                         MADELINE  ORGRATZI                   $39,000.00
                PERSONNEL                         CYNTHIA   JOHNSON                    $13,500.00
                PERSONNEL                         TOM       FITZHUGH                   $13,000.00

                                                                                       $145,500.00


                PUBLIC RELATIONS                  MONTE     BANK                       $80,000.00
                PUBLIC RELATIONS                  LAURA     PENMAN                     $39,000.00
                PUBLIC RELATIONS                  CHARLES   BOWER                      $38,500.00
                PUBLIC RELATIONS                  BETSY     ZEDI                       $37,000.00
                PUBLIC RELATIONS                  JOCK      JACKSON                    $34,000.00
                PUBLIC RELATIONS                  MICHAEL   ANGELO                     $18,000.00
                PUBLIC RELATIONS                  CAROL     MCDOUGALL                  $18,000.00

                                                                                       $264,500.00


                THERMOREGULATION                  ROGER     WILCO                      $80,000.00
                THERMOREGULATION                  PHINEAS   FINN                       $45,000.00
                THERMOREGULATION                  TERRY     CLOTH                      $38,000.00
                THERMOREGULATION                  MARK      TIME                       $33,000.00
                THERMOREGULATION                  JOE       KASPAR                     $31,000.00

                                                                                       $227,000.00

                                                                                     $2,522,500.00
```

Example 3

```
INPUT 15000 F 30000 PS DD=SYS010
REC PART1          1  13000
REC PART2      13001   1999
01OUTPUT 15100  D  PS DD=SYS020
01510001 PART1
0151     'TECHNICAL RESEARCH'  COLUMN=13001
0151     PART2                 COLUMN=13021
```

This example illustrates how to insert a field into a long record. The input is a sequential file assigned to ddname SYS010 in the z/OS JCL. The input file contains fixed-length 15,000-byte records.

The OUTPUT parameter defines a sequential output file that is to contain detail records 15,100 bytes in length. The edit parameters define the records. The output records contain:

- The first part of the input record in columns 1 through 13,000.
- The literal 'TECHNICAL RESEARCH' starting in column 13,001; since the column number extend beyond 9999, the COLUMN= keyword expression must be used.
- The second part of the input record starting in column 13,021.

Chapter 5: Process Parameters

This section contains the following topics:

Overview

Process parameters define procedure logic for processing data. These parameters are of two types:

Type	What it does
7	Execute during the extract phase of CA Culprit processing and are applied each time a record or group of records is delivered to the input buffer.
8	Execute during the output phase of CA Culprit processing and are applied to system-maintained totals of extracted items.

Topics

The following topics are discussed in this chapter (followed by examples of process parameters):

- **User-defined variables** defines coding rules for field names and user-supplied constants used in type 7 and type 8 procedure logic.
- **Process parameter syntax** supplies syntax and syntax rules for process parameters.
- **Process operations** supplies syntax and syntax rules for arithmetic, conditional, assignment, and control operations.

User-Defined Variables

Field name values and user-supplied constants are used in type 7 and type 8 procedure logic. These variables are defined on the following pages.

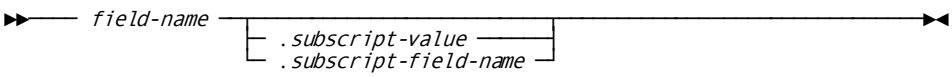
Field-name-expression

Purpose

Specifies an occurrence of a singly- or multiply-occurring field defined on a REC parameter or work field parameter.

The field can be defined as alphanumeric or numeric. The data in a numeric field can be in any numeric format; that is, binary, zoned decimal, packed signed decimal, packed unsigned decimal, or bit.

Syntax



Syntax Rules

field-name

Specifies the name of a singly- or multiply-occurring field defined on a REC or work field parameter.

subscript-value/subscript-field-name

Specifies a subscript for a multiply-occurring input or work field.

If a subscript value is provided, it must be separated from the name of the multiply-occurring field by a period (.). The value of the subscript should be an integer in the range 1 to *n*, where *n* is the number of repetitions of the multiply-occurring field.

For example, if EMPLOYEE occurs ten times, the value of the subscript should not exceed 10. The value of the subscript can be tested in procedure logic.

subscript-value

Specifies a numeric literal that identifies a specific occurrence of a multiply-occurring field; for example, EMPLOYEE.4 identifies the fourth occurrence of EMPLOYEE.

subscript-field-name

Specifies the name of a singly-occurring numeric input or work field whose value identifies a specific occurrence of a multiply-occurring field; for example, if INDX has a value of 2, then EMPLOYEE.INDX identifies the second occurrence of EMPLOYEE.

The subscript field definition must not specify a decimal point.

Usage

The following CA Culprit reserved words can replace *field-name-expression* in procedure logic:

- **M*ID** applies exclusively to match-file runs and can be specified in type 7 procedure logic only. M*ID is used in procedure logic to test the status of input files in a match-file run.

  In a match-file run, CA Culprit creates a 1-byte binary field for each file; the field is called a file-specific status byte. M*ID is a binary combination of the file-specific status bytes for all files in a match run. This field can be tested for values that reflect various conditions, as listed in the following table.

Decimal Value of M*ID	Condition
8	File(s) out of sequence
4	Duplicate key value
-	Not used
1	End of file(s)

Note: For more information on match-file runs, see the *CA Culprit for CA IDMS User Guide*.

- **LEVL** refers to the current level of a control break coded on a SORT parameter. It can be specified in type 8 procedure logic only.

  LEVL is used to test for a specific control break; if the test is true, CA Culprit executes procedure logic specified for the control break.

  LEVL can be equated to values in the range 1 through 21. For the most minor control break, LEVL equals 1; at grand totals time, LEVL equals one more than the number of control break codes specified on the SORT parameter.

More information:

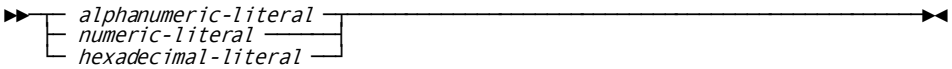
Output Definition Parameters (see page 69)

User-supplied Constant

Purpose

An alphanumeric, numeric, or hexadecimal literal used to represent a value.

Syntax



Syntax Rules

alphanumeric-literal

A 1- to 64-character string enclosed in single quotation marks that consists of letters, numbers, and special characters in any combination.

To express an apostrophe (a single quotation mark), code two consecutive single quotation marks; for example, 'ERNIE''S DINER'.

numeric-literal

A 1- to 31-digit numeric value, optionally preceded by a sign and optionally containing an embedded or trailing decimal point; for example, 12.34.

hexadecimal-literal

A 1- to 50-character (25-byte) hexadecimal value, preceded by X and enclosed in single quotation marks; for example, X'0000'.

Process Parameter

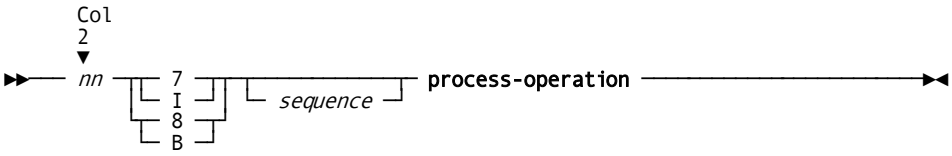
Purpose

Specifies arithmetic, conditional, assignment, and control operations.

The first seven columns on a process parameter follow a fixed format. Process operations coded in column 8 or later follow a free format.

Because several options exist for coding process parameters, only the syntax associated with the fixed portion of the parameter is discussed in detail here. Syntax associated with arithmetic, conditional, assignment, and control operations is discussed under Process Operations.

Syntax



Syntax Rules

nn

Identifies the report associated with the process parameter. *Nn* must be a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

7

A single character coded in column 4 to indicate input processing; alternatively, I (input) can be used.

Input processing is performed during the extract phase. Logic defined on type 7 parameters is executed each time a record or group of records is delivered to the input buffer.

Type 7 parameters are used for such functions as eliminating records from further processing in a specific report and performing arithmetic operations on input fields.

8

A single character coded in column 4 to indicate break processing; alternatively, B (break) can be used. Break processing is performed during the output phase. Logic defined on type 8 parameters is executed when a control break is encountered.

Type 8 parameters are used for such functions as checking the control-break level, selecting a literal for use as a label in a total line, and manipulating system-maintained totals of extracted items.

sequence

Defines the order of this procedure statement relative to other process parameters of the same type. *Sequence* is a 1- to 3-digit right-justified number coded in columns 5 through 7. Leading zeros can be omitted. If a sequence number is not specified, columns 5 through 7 must be blank.

Sequence numbers are optional unless they are needed to indicate the object of a conditional or control operation. Process parameters without sequence numbers must be entered in the order in which they are to execute.

In the parameter sort phase, CA Culprit sorts the process parameters in order of ascending sequence number. Unsequenced process parameters that immediately follow a sequenced process parameter maintain their position relative to the sequenced process parameter.

process-operation

A value or keyword that indicates an arithmetic, conditional, assignment, or control operation.

Syntax and rules for process operations within these categories are presented under Process Operations.

More information:

Introduction (see page 13)
Output Definition Parameters (see page 69)

Process Operations

Types of Operation

Four major types of instructions can be coded on process parameters, as follows:

- **Arithmetic operations** add, subtract, multiply, or divide field or literal values by means of simple or compound operations.
- **Conditional operations** compare the contents of one field or literal value with the contents of another field or literal value. When the test condition is true, the specified action occurs.
- **Assignment operations** move a field or literal value to a work field or an internal CA Culprit field.
- **Control operations** control the flow of processing in the extract and output phases.

Each type of process operation is discussed separately below. The following table identifies CA Culprit keywords associated with process operations and summarizes their function.

Process Operation	Keyword Command	Action
Arithmetic	ADD	Adds
	MINUS	Subtracts
	TIMES	Multiplies
	DIVIDE	Divides
	COMPUTE	Performs a compound arithmetic operation
Conditional	IF	Performs a compound test operation
	EOF	Tests for an end-of-file condition

Process Operation	Keyword Command	Action
	B	Performs an unconditional branch
Assignment	MOVE	Transfers a value to another field
	CONVERT	Converts an alphanumeric value to a numeric value
Control	PERFORM	Performs an unconditional branch
	RETURN	Returns to point of branch specified on a PERFORM statement
	CALL	Calls an external procedure module
	PICK	Selects edit lines for extracting or printing
	UNPICK	Deselects edit lines for extracting or printing
	TAKE	Prints selected edit lines; processing for the report ceases
	RELS	Prints selected edit lines; processing for the report continues
	DROP	Does not print selected edit lines; processing for the report ceases

Overview of Arithmetic Operations

Arithmetic operations perform addition, subtraction, multiplication, and division. A simple arithmetic operation involves one operation; a compound arithmetic operation involves two or more operations.

When a compound arithmetic operation is specified on a process parameter, CA Culprit breaks down the compound operation into a series of simple arithmetic operations. Each of these system-generated statements is printed on the Input Parameter Listing, as shown on the following page. CA Culprit uses the following criteria to resolve compound arithmetic operations into simple arithmetic operations:

- Operations enclosed in parentheses are performed first.
- Multiplication and division are performed next, in order of occurrence from left to right.
- Addition and subtraction are performed last, in order of occurrence from left to right.

When a compound expression is resolved into multiple simple arithmetic operations, intermediate results are stored in internal work fields. These internal work fields are defined and maintained by CA Culprit. The following figure illustrates the internal work-field names CA Culprit uses in these operations.

When fields with a varying number of decimal positions are involved in arithmetic operations, intermediate results are computed to the precision of the field containing the most decimal positions. When the ROUND option is in effect (see ROUND later in this chapter), intermediate results are computed to two extra decimal positions for additional accuracy. After the number of decimal positions is determined for internal result fields, the simple arithmetic operations generated by the COMPUTE statement are performed.

Only the result field value changes in an arithmetic operation. The values of the left and right operands remain unchanged, unless one of these fields is also the result field.

Input Parameter Listing of a Compound Arithmetic Operation:

```
mm/dd/yy                  INPUT PARAMETER LISTING       Vnn.n  PAGE     2


********
PROCESS USER   INTERNAL
        LABEL  SEQUENCE     PROCESS STATEMENT
********
 01  8  010        1    $   COMPUTE (SALARY + COUNT) X COUNT - INDX RESULT  <---- User input
 01  8             2    SALARY    +         COUNT     WORK*01 -,
 01  8             3    WORK*01   X         COUNT     WORK*01  |-- CA-Culprit's interpretation of
 01  8             4    WORK*01   -         INDX      RESULT  -'   the compound arithmetic expression
```

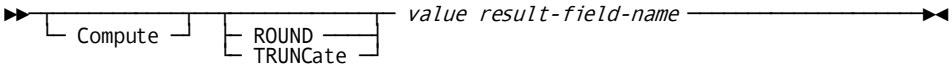
Note: Compound arithmetic operations appear as simple arithmetic statements on the Input Parameter Listing. Internal work fields (WORK*01) store intermediate results.

Arithmetic Operations

Purpose

Perform addition, multiplication, subtraction, and division.

Syntax



Syntax Rules

Compute

Identifies a compound arithmetic operation; it is an optional keyword for simple arithmetic operations unless either parentheses enclose the arithmetic operation or the ROUND/TRUNCATE clause is specified.

ROUND

Rounds the resulting value to the number of decimal positions defined for the result field. CA Culprit rounds up if the value is .5 or greater; for example, 0.5 is rounded to 1.

TRUNCate

Truncates the resulting value to the number of decimal positions defined for the result field.

If one of these options is specified, it must follow the keyword COMPUTE.

ROUND is the default when PROFILE option RELEASE=6 is in effect; TRUNCATE is the default when PROFILE option RELEASE=5 is in effect.

value

Specifies an arithmetic operation that contains operands and operators, as follows:

- An **operand** must be a numeric literal (*literal*) or numeric field name (**field-name-expression**). The left operand is separated from the right operand by an arithmetic operator (see below).

  An arithmetic expression that contains two operands connected by an operator is a simple arithmetic expression, for which the keyword COMPUTE is optional. An arithmetic expression that contains more than two operands is a compound arithmetic expression, for which the keyword COMPUTE is required.

- An **operator** indicates the type of arithmetic calculation:

Operator	Synonym	Meaning
+	ADD	Indicates that the value of the left operand is added to the value of the right operand.
-	S or MINUS	Indicates that the value of the right operand is subtracted from the value of the left operand.
X	TIMES	Indicates that the value of the left operand is multiplied by the value of the right operand.
/	DIVIDE	Indicates that the value of the left operand is divided by the value of the right operand. If the value of the right operand is zero, a divide exception occurs and the result of the operation is set to zero.

Nonnumeric data and arithmetic overflows cause error messages. Arithmetic overflows are reported by the extended error-handling facility.

Note: For more information, refer to the *CA Culprit for CA IDMS Messages and Codes Guide*.

result-field-name

The name of a numeric work field that receives the result of the arithmetic operation. *Result-field-name* must be defined on a work field parameter.

More information:

Release 5 and 6 Default Actions (see page 365)
PROFILE Parameter (see page 29)

Overview of Conditional Operations

Conditional operations define test conditions that compare values held by different fields. When the test condition is true for the input record or control break being processed, the specified action occurs. Otherwise, the specified action is ignored, and control passes to the next processing statement.

Types of Conditional Operations

CA Culprit can perform the following types of conditional operations:

- **Test operations** compare one or more pairs of values by means of a test operator and AND and OR connectors. A simple test operation compares two values; a compound test operation compares more than two values.

  When a compound test operation is specified on a process parameter, CA Culprit breaks down the compound operation into a series of simple test operations. Each of these system-generated statements is printed on the Input Parameter Listing, as shown in the following figure. CA Culprit uses the following criteria to resolve compound test operations into simple test operations:

  - Operations enclosed in parentheses are performed first.
  - Operations connected by AND are performed next, in order from left to right.
  - Operations connected by OR are performed last, in order from left to right.

- **End-of-file (EOF) operations** test for an end-of-file condition. When the end of the input file is reached, type 7 procedure logic is reentered and processed a final time until a DROP, TAKE, or STOP-RUN instruction executes. EOF operations are appropriate for any kind of input file or database.

- **Branch operations** unconditionally transfer processing control to a specified process parameter or specified result action.

Input Parameter Listing of a Compound Test Operation:

```
mm/dd/yy                    INPUT PARAMETER LISTING        Vnn.n  PAGE     3
********
PROCESS USER   INTERNAL
        LABEL  SEQUENCE     PROCESS STATEMENT
********
 01  7  010        1    $   IF ((SALARY GT 10000)                                    ┐
 01  7                   $         AND (TITLE NE 'DATA ENTRY CLERK    '))             ├── User input
 01  7                   $         OR EMP-LNAME EQ ('BREEZE         ' TO 'LANCHESTER     ') │
 01  7                   $         TAKE                                               ┘
 01  7             2    SALARY    GT        10000     INTSEQ-4              ──┐
 01  7             3    B         INTSEQ-5                                  │  CA-Culprit's interpretation
 01  7             4    TITLE     NE        'DATA ENTRY CLERK    '    TAKE  ├── of the compound test
 01  7             5    EMP-LNAME GE        'BREEZE         '    INTSEQ-7  │  operation
 01  7             6    B         INTSEQ-8                                  │
 01  7             7    EMP-LNAME LE        'LANCHESTER     '    TAKE      │
 01  7             8    $                                                 ──┘
```

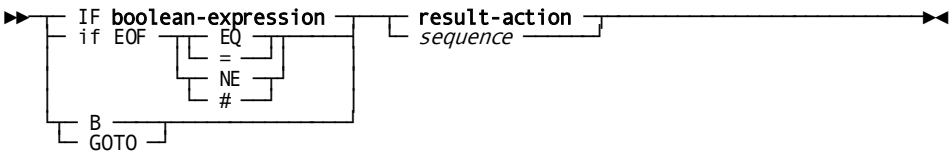
Note: CA-Culprit generates simple test operation statements from a compound test operation statement; CA-Culprit also generates branch instructions to internal sequence numbers (INTSEQ-4) in order to preserve the logic of the original expression.

Conditional Operations

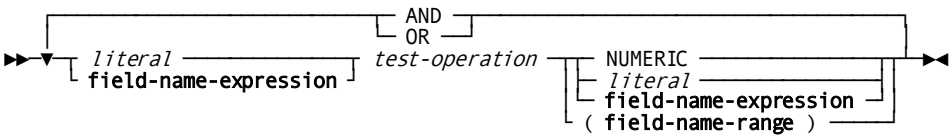
Purpose

Define what action to take based on a comparison of values held by different fields.

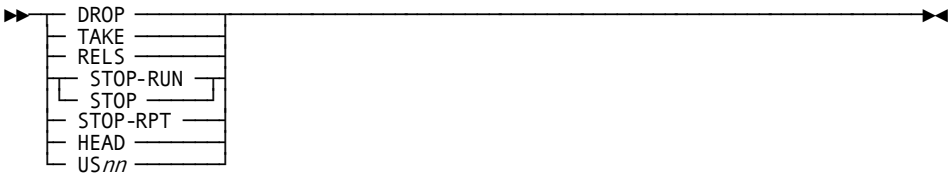
Syntax



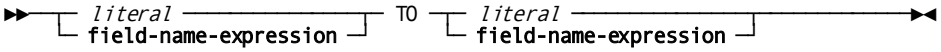
Expansion of Boolean-expression



Expansion of Result-action



Expansion of Field-name-range



Syntax Rules

IF boolean-expression

Specifies a test operation. **IF** is a required CA Culprit reserved word that indicates a compound test operation. IF is an optional keyword for a simple test operation unless parentheses enclose the operation. For **boolean-expression**, see the explanation of the expanded syntax below.

if EOF EQ/NE

Specifies a test for an end-of-file condition.

This test is valid only on a type 7 process parameter. EQ/NE indicates the type of test to be performed:

Test	What it means
Eq	Indicates that the specified action occurs if end-of-file is reached.
Ne	Indicates that the specified action occurs if end-of-file has not been reached.

A STOP-RUN or a STOP-RPT instruction cannot be specified as a result action on an EOF conditional operation.

When end-of-file is reached, the procedure logic is evaluated a final time, starting with the first procedure statement. Processing continues until a DROP, TAKE, or STOP-RUN instruction executes. Once one of these instructions executes, the current buffer does not reenter the type 7 procedure logic for the report.

B

Directs CA Culprit to perform an unconditional branch.

result-action

Indicates the action to be performed as a result of a true test condition or an unconditional branch statement.

See expanded syntax below for possible values for **result-action**.

sequence

The sequence number of a process parameter that receives processing control. Processing control must be passed to a process parameter of the same type; For example, from one type 7 parameter to another or from one type 8 parameter to another.

The process statement that receives processing control must have a sequence number coded in columns 5 through 7.

Expansion of Boolean-expression

***literal*/field-name-expression**

Specifies the value of the left operand.

Syntax and syntax rules for coding **field-name-expression** appear in User-Defined Variables. The left and right operands of a test operation must contain data that is compatible for comparison. The following table presents combinations of data types that can be compared in a test operation.

Alphanumeric data and numeric data are compared as follows:

- Alphanumeric operands are compared starting with the leftmost character of the character string. Excess characters on the right of the longer value are ignored (for example, 'ABCDE' EQ 'ABC' produces a true condition). A warning message is printed in the Input Parameter Listing when this occurs; excess characters are ignored.
- Numeric operands are compared by true decimal value regardless of field length. The field with fewer decimal positions is padded with zeros (for example, 65.3 LT 98.095 is compared as if the values were 65.300 and 98.095).

Valid Left and Right Operand Data Types and Associated Test Operations

Left Operand	Valid Test Operators*		Right Operand
	EQ/NE	GT/LT/GE/LE	
Numeric literal	●	●	Numeric literal
	●	●	Numeric input or work field
Numeric input or work field	●	●	Numeric literal
	●	●	Numeric input or work field
	●		NUMERIC (with input field only)

Left Operand	Valid Test Operators*		Right Operand
Alphanumeric or hexadecimal literal	●	●	Alphanumeric or hexadecimal literal
	●	●	Alphanumeric or hexadecimal input or work field
Alphanumeric input or work field	●	●	Alphanumeric or hexadecimal literal
	●	●	Alphanumeric or hexadecimal input or work field
LEVL	●	●	Numeric literal
EOF	●		None

Note:

```
* EQ equal to
  GT greater than
  GE greater than or equal to
  NE not equal to
  LT less than
  LE less than or equal to
```

test-operation

Indicates the type of test to be performed:

Symbol	Synonym	What it means
EQ	E or=	Indicates the values of the left and right operands are equal.
NE	N or #	Indicates the values of the left and right operands are not equal.
GT	H or >	Indicates the value of the left operand is greater than the value of the right operand.
LT	L or <	Indicates the value of the left operand is less than the value of the right operand.
GE	>= or =>	Indicates the value of the left operand is greater than or equal to the value of the right operand.
LE	<= or =<	Indicates the value of the left operand is less than or equal to the value of the right operand.

The test must be compatible with the type of data contained in the left and right operands; possible combinations of data types and test operations are presented in the Valid Left and Right Operand Data Types and Associated Test Operations table.

A maximum of 32 test conditions can be defined on a single process parameter. One or more test conditions can be enclosed in parentheses. CA Culprit evaluates expressions within parentheses before evaluating expressions not enclosed in parentheses.

NUMERIC

A keyword value for the right operand. NUMERIC indicates that the value of the left operand is to be tested for numeric data. Valid test operations with this keyword are EQ and NE. An EQ test is true if the value of the left operand is numeric; a NE test is true if the value of the left operand is not numeric. If nonnumeric data is discovered, the record that contains the nonnumeric data can be dropped from further processing to avoid a data exception.

***literal*/field-name-expression**

Specifies either a single numeric literal or field name or a list of numeric literals or field names for comparison, as follows:

- A single value identifies a simple test operation for which the keyword IF is optional.
- A list of values identifies a compound test operation for which the keyword IF is required. The list must be enclosed in parentheses; a comma or space must separate one listed item from another.

  Valid test operators for a list of items are EQ and NE. An EQ test operator implies an OR connector between the listed values; a NE test operator implies an AND connector between the listed values.

Syntax and syntax rules for coding *field-name-expression* appear in User-Defined Variables.

field-name-range

Identifies a range of values in the right operand to be compared to the left operand. The range of values must be enclosed in parentheses; the keyword TO must join the first and last values of the range.

Valid test operators for this expression are EQ and NE. An EQ test operation is true if the value of the left operand equals some value in the specified range, inclusive. A NE test operation is true if the value of the left operand is not equal to any value in the specified range, inclusive.

See expanded syntax diagram above for details of **field-name-range**.

AND/OR

Logically connects two test operations, as follows:

- **AND** specifies that the test condition is true only if both operations are true.
- **OR** specifies that the test condition is true if either one of the operations is true.

Expansion of Result-action

In this discussion, the words **buffer** and **control break** are used as follows:

- **Buffer** refers to the record or group of records contained in the input buffer. The input buffer is processed during the extract phase. This phase performs procedure logic coded on type 7 process parameters for every report in the run. This phase also extracts detail line information.
- **Control break** refers to a control break processed during the output phase. This phase performs procedure logic coded on type 8 process parameters and processes header and total lines for output.

DROP

Indicates that processing for the current buffer or control break is complete for the report associated with this parameter. Any remaining procedure logic defined for this report is not executed; the code is not accessed again until the next buffer or control break is processed. Once a DROP executes, no additional edit lines are extracted or printed for the current buffer or control break for this report. Information that was output before the DROP instruction is not affected.

TAKE

Indicates that processing for the current buffer or control break is complete for the report associated with this parameter. Edit lines selected for the buffer or for the control break are extracted or output. Any remaining procedure code defined for the report is not executed for the current buffer or control break.

RELS

Indicates that edit lines selected for the current buffer or for the current control break are extracted for output. The remaining procedure code defined for the report is processed. Processing control returns to the procedure statement that immediately follows the RELS instruction.

Any number of RELS statements can be issued for each buffer or control break by each report. The RELS instruction allows information to be extracted or printed for a buffer or control break and allows the buffer or control break to be processed further.

STOP-RUN/STOP

Indicates that processing for the current buffer is complete for the report associated with this parameter; a STOP-RUN instruction in type 7 logic forces an end-of-file condition. All reports in the run finish processing the current buffer. If an EOF test is included anywhere in the run, any logic associated with the test is then executed.

STOP-RUN can appear only on a type 7 process parameter; it must not be coded as the result action of an EOF test. Since all reports processed by a CA Culprit run are affected by a STOP-RUN statement, a warning message is printed in the Input Parameter Listing.

STOP may be used as a synonym for STOP-RUN.

STOP-RPT

Indicates that processing for the current buffer is complete for the report associated with this parameter and that no further input records are to be processed for this report. Type 7 logic for this report is entered again only when an end-of-file condition is reached and only if an EOF procedure statement is coded for the report. Type 7 logic for other reports in the run continues to execute.

STOP-RPT can appear only on type 7 process parameters; it must not be the result action of an EOF procedure statement.

HEAD

Causes a special extracted-items record to be written to the extracted items and statistics file; the record contains the current values for all variables, other than sort keys, that are referenced on type 4 edit parameters. HEAD can appear only on a type 7 process parameter.

Values for variable header fields that are not sort keys must be extracted by specifying HEAD as a result action. A HEAD instruction generates additional procedure code and causes special records to be written to the extracted items file. Therefore, it is more efficient to include all variable fields in type 4 edit parameters as sort keys.

Records extracted by a HEAD instruction are sorted in accordance with the values of the related sort-key fields, but ahead of detail lines with the same sort-key values. Header variables extracted by a HEAD instruction are inserted into appropriate header lines in the output phase.

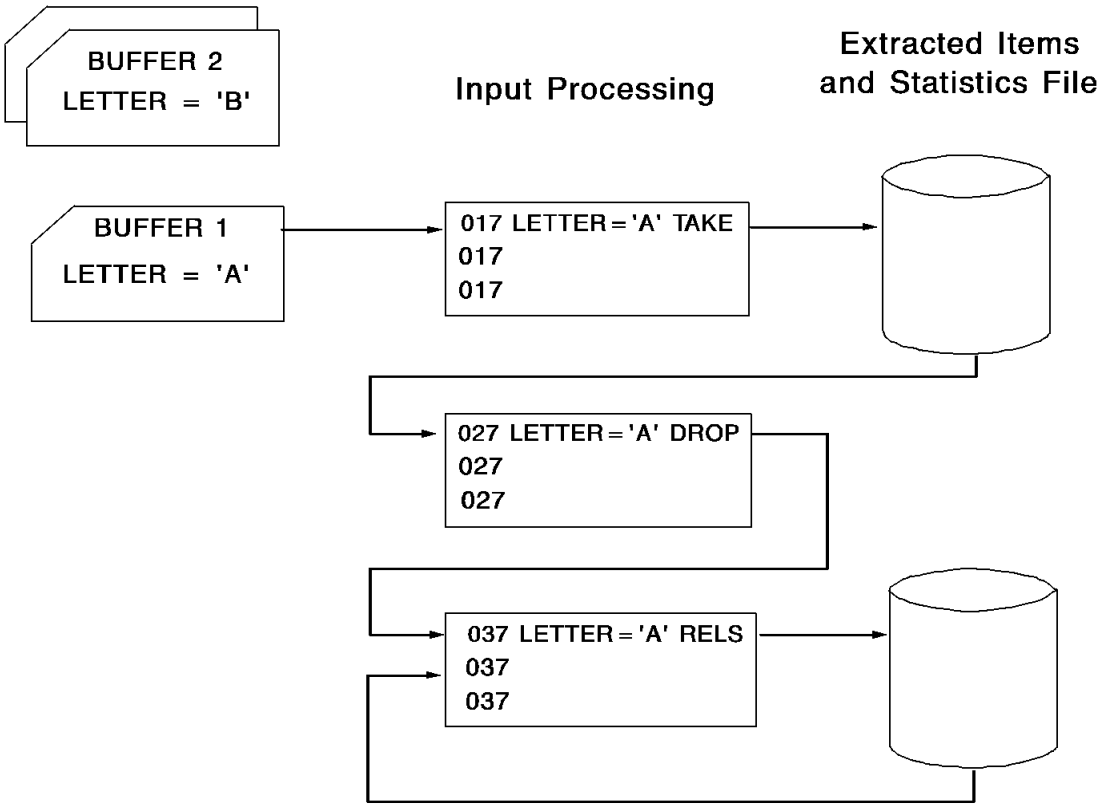
US*nn*

Indicates that processing control is passed to a user-coded or CA-supplied procedure module. *Nn* is a 2-digit number in the range 00 through 99. US*nn* can be specified only on a type 7 process parameter that contains a B (branch) or CALL instruction; the B instruction is discussed earlier; the CALL instruction is discussed under Control Operations (see page 123).

After the named procedure completes execution, processing control returns to the procedure statement immediately following the statement that contains US*nn*.

Note: For more information on CA Culprit procedure modules, see the *CA Culprit for CA IDMS User Modules Guide*.

The following figure illustrates how DROP, TAKE, and RELS instructions differ in procedure logic. This figure applies only to type 7 procedure logic; in type 8 procedure logic, the instructions would be processed at the end of the output phase and at each control break.



Note: This figure illustrates CA Culprit processing for one input buffer in type 7 procedure logic.

More information:

Control Operations (see page 123)

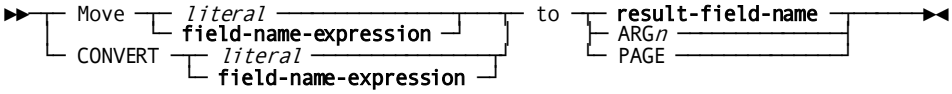
Assignment Operations

Purpose

Assign a value contained in a literal or variable field to either a work field or a CA Culprit reserved field.

MOVE operations transfer information to a work field or to the CA Culprit reserved fields PAGE and ARG*n*. CONVERT operations convert an alphanumeric value to a numeric value stored in a work field.

Syntax



Syntax Rules

Move *literal*/field-name-expression

Specifies that a MOVE operation is to be performed. *Literal*/**field-name-expression** identifies the value to be moved; syntax and syntax rules for coding **field-name-expression** appear in User-Defined Variables (see page 104) earlier in this chapter.

The following considerations apply to alphanumeric and numeric values; CA Culprit treats hexadecimal values as alphanumeric:

- When an **alphanumeric** field or literal is moved to a result field of a different length, the following actions occur:
  - If the field or literal that is moved contains more characters than the result field, excess characters are truncated on the right.
  - If the result field contains more characters than the field or literal that is moved, the moved field is left-justified in the result field and blank spaces fill out the right side of the result field. If truncation or padding occurs, CA Culprit issues a warning message.
- When a **numeric** field or literal is moved to a result field that does not have the same length or number of decimal positions, the following actions occur:
  - If a numeric field is moved to a result field that contains more decimal positions, the low-order decimal positions are padded with zeros.

- If a numeric field is moved to a result field that contains fewer decimal positions, the result is rounded to the number of decimal places in the result work field.
- If a numeric result field cannot accommodate all significant digits of the moved field, significant digits are truncated on the left and an arithmetic overflow results. Arithmetic overflows are reported by the extended error-handling facility.

  Note: For more information, see the CA Culprit for CA IDMS Messages and Codes Guide.

PROFILE parameter options RELEASE=5 and RELEASE=6 do not handle situations that involve different numbers of characters or decimal positions in the same manner. See Error! Reference source not found., for a complete discussion of these differences.

CONVERT *literal*/field-name-expression

Specifies that a CONVERT operation is to be performed.

Literal/**field-name-expression** identifies an alphanumeric literal, work field, or input field whose value is to be converted into a numeric value. Syntax and syntax rules for coding **field-name-expression** appear in User-Defined Variables (see page 104).

Considerations for coding *literal* appear below:

- Leading blanks are permitted, but the entire field must not be blank; leading blanks are treated as if they were zeros.
- A minus sign is accepted in the first (leftmost) position of the field or literal if the value is negative.
- A leading, trailing, or embedded decimal point is permitted.
- Any characters outside the range 0 through 9 produce unpredictable results.

  When an integer alphanumeric value is converted into an integer numeric value, the result is truncated or padded on the left. If a decimal point is specified for the alphanumeric value or the numeric value, the following occurs:

- If the alphanumeric value contains more decimal positions than are indicated for the result field, the result is rounded.
- If the alphanumeric value contains fewer decimal positions than are indicated for the result field, the result is padded with low-order zeros.
- If the result field is too small to hold all the significant digits, significant digits are truncated on the left and an arithmetic overflow results. Arithmetic overflows are reported by the extended error-handling facility.

  Note: For more information, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

result-field-name

The name of a work field that receives the result of the MOVE or CONVERT operation.

In a MOVE operation, the data type of a sending field or literal must be compatible with the data type of the result field; that is, both fields must be defined as numeric or as alphanumeric. Hexadecimal literals are treated as alphanumeric values for the purposes of a MOVE operation.

In a CONVERT operation, the data type of the result field must be numeric.

ARG*n*

A CA Culprit reserved word that passes information to a user procedure module.

N must be an integer in the range 1 through 9. ARG*n* is valid only for MOVE operations and can appear only on type 7 process parameters.

Note: For more information on user procedure modules, see the *CA Culprit for CA IDMS User Modules Guide*.

PAGE

A system-defined numeric work field that receives a report page number.

This reserved word is valid only for MOVE operations and can appear only on type 8 process parameters. When PAGE is used in a MOVE operation, the report page number is reset. The specified value plus 1 is printed as the next page number. The value of PAGE can be printed on each page of the associated report on either the title line or a header line if PAGE is referenced on a type 4 edit parameter.

Control Operations

Control operations manage the flow of processing in a CA Culprit report. The following control operations are available:

- **PERFORM** passes processing control to code in another area of procedure logic. A PERFORM instruction is useful when a group of procedure statements may execute any number of times. Fewer statements are required, and procedures that are executed repeatedly can be centralized.
- **RETURN** returns processing control to the statement that immediately follows PERFORM. The RETURN statement executes only if the procedure code is accessed with a PERFORM statement. Procedure code accessed by a PERFORM instruction can also be accessed inline (that is, by falling through the code) or by coding a branch instruction to the first statement of code.

- **CALL** calls an external user-coded or CA-supplied procedure module to be executed. The CALL statement is the equivalent of coding individual MOVE ARG*n* statements that first establish the arguments in the argument list and then coding a B US*nn* statement to invoke the procedure module. The CALL statement generates appropriate MOVE and branch instructions. These instructions appear on the Input Parameter Listing. Details on coding MOVE and branch instructions are presented earlier in this chapter.
- **PICK, UNPICK, TAKE, RELS, and DROP** select type 5 edit lines for extracting from a particular input buffer or type 6 edit lines for output from a particular control break. All type 5 and type 6 edit parameters are initially given the same select/deselect status at the start of each report; the status is reset each time a new buffer or control break is processed. All type 5 and type 6 edit parameters are initially set to a status of select unless a PICK, TAKE, or RELS instruction specifies lines for output, in which case they are initially set to a status of deselect.

  Output lines are sorted as follows:

  - Report number
  - Sort keys specified on the SORT parameter
  - Header or detail data indicator (data derived from a branch to HEAD sorts first)
  - Input buffer number
  - Edit line number
  - Release count (a counter maintained by CA Culprit that is set to zero when procedure logic is entered for each report and incremented by 1 for each RELS, TAKE, or DROP executed)

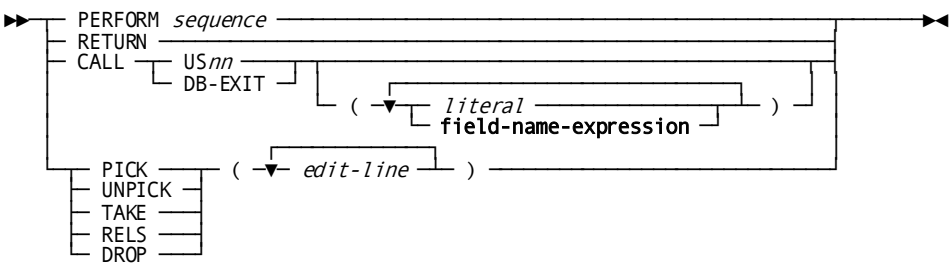
  A special user-defined sort key is required to print lines outside of their numerically defined order; for example, RELS line 5, then RELS line 1. Any type of work field can be coded as a sort key to define the desired sort sequence on the SORT parameter; this key must follow all report-related sort fields. Before a record is extracted by means of a RELS statement, the user can move a print sequence value to this work field. CA Culprit sorts all extracted records on this value; records with the same value sort by edit line number.

Control Operations

Purpose

Control the flow of processing.

Syntax



Syntax Rules

PERFORM

Passes processing control to a process parameter of the same type; for example, from one type 7 parameter to another or from one type 8 parameter to another.

sequence

The sequence number of the process parameter that receives processing control. The process parameter that receives processing control must have the sequence number coded in columns 5 through 7.

PERFORM instructions cannot be nested; that is, a PERFORM statement cannot appear within another PERFORM statement. All CA Culprit procedure statements other than PERFORM, however, can appear within code accessed by a PERFORM instruction.

RETURN

Returns processing control to the process statement that immediately follows the statement containing the PERFORM instruction. RETURN executes only if the procedure code is accessed with a PERFORM statement.

CALL

Indicates that processing control is passed to a user-coded or CA-supplied procedure module.

CALL instructions must be coded only on type 7 process parameters.

Note: For more information on coding CALL statements and syntax for individual procedure modules, see the *CA Culprit for CA IDMS User Modules Guide*.

US*nn*

Identifies the name of the external procedure module that is to receive processing control. *Nn* must be a 2-digit number in the range 00 through 99.

***literal*/field-name-expression**

Indicates one or more literals, input fields, or work fields whose values serve as arguments for the procedure module identified by US*nn*. Syntax and syntax rules for coding **field-name-expression** appear under User-Defined Variables (see page 104). These literals or field values are moved to ARG*n* fields, which communicate with the procedure module.

Note: For more information on coding arguments for individual procedure modules, see the *CA Culprit for CA IDMS User Modules Guide*.

A maximum of nine arguments can be specified. These arguments must be specified in sequence; that is, ARG1 corresponds to the first literal or field in the list, ARG2 corresponds to the second argument in the list, and so forth.

Arguments cannot be omitted; for example, if a procedure module requires only the first and third arguments, a dummy value must be specified for the second argument. A blank can serve as a dummy value for alphanumeric arguments; a zero can serve as a dummy value for numeric arguments.

PICK/UNPICK/TAKE/RELS/DROP ***(edit-line)***

Specifies edit line selection/deselection operations as described below. These operations allow the user to specify which edit lines to print. When these instructions appear on a type 7 process parameter, detail lines are selected; when these instructions appear on a type 8 process parameter, total lines are selected.

PICK

Selects edit lines for extracting or printing. These lines are not output, however, until a TAKE or RELS instruction is executed. Multiple PICK statements can execute before the selected lines are output.

UNPICK

Deselects edit lines for extracting or printing. Edit lines not specified on an UNPICK instruction are selected for output; the selected lines are not extracted or output until a TAKE or RELS instruction executes.

TAKE

Immediately extracts or outputs the selected edit lines, and stops processing for the current buffer or control break. Any remaining procedure code defined for the report does not execute.

An implicit TAKE instruction executes at the end of type 7 and 8 procedure logic thereby causing all selected edit lines to be extracted or output. If no explicit selection or deselection has occurred, all edit lines are output.

RELS

Immediately extracts or outputs the selected edit lines and continues processing for the current buffer or control break. Processing continues with the procedure statement that immediately follows the RELS instruction. Any number of RELS instructions can be issued for each buffer or control break by each report.

Note: The selected/deselected status is not reset following a RELS instruction, since the current buffer or control break continues to be processed. PICK or UNPICK instructions are needed to ensure that the necessary lines are selected for any subsequent TAKE or RELS instructions, unless the TAKE and RELS instructions specify edit lines to be selected.

DROP

Indicates that processing control for the current buffer or control break is complete for the report associated with this parameter. Any remaining procedure logic defined for this report is not executed; the code is not accessed again until the next buffer or control break is processed.

Once a DROP executes, no additional edit lines are extracted or output for the current buffer or control break for this report. Information extracted or output previous to a DROP instruction is unaffected.

edit-line

Identifies an edit line or a list of edit lines to be selected or deselected. *Edit-line* is a number in the range 1 through 8 that refers to the value coded in position 5 on a type 5 or a type 6 edit parameter.

If a list of edit line numbers is specified, the list must be enclosed in parentheses; each value in the list must be separated by a comma or a space.

An edit line specification is required for PICK and UNPICK instructions; it is optional for TAKE, RELS, and DROP instructions. If no edit lines are specified on a TAKE or RELS instruction, all lines selected by PICK or UNPICK instructions are extracted or output; if no edit lines are explicitly selected or deselected, all edit lines are extracted or output when a TAKE or RELS instruction executes.

Note: TAKE, RELS, and DROP are optional actions that can be performed as the result of a conditional operation discussed earlier in this chapter. However, when used in this context, edit lines cannot be specified for selection or deselection.

Examples

Sample process parameters are shown and described below.

Example 1: Arithmetic Operation—Simple

```
017      PREVIOUS-BALANCE + DEPOSITS CURRENT-BALANCE
```

For each record processed for Report 01, the value in DEPOSITS is added to the value in PREVIOUS-BALANCE; the sum is stored in the work field CURRENT-BALANCE. The values of PREVIOUS-BALANCE, DEPOSITS, and CURRENT-BALANCE for each input buffer processed are available for output on detail lines.

Example 2: Arithmetic Operation—Compound

```
028010 COMPUTE (REVENUE - COST) / COUNT AVE-PROFIT
```

This procedure statement executes in the output phase; because a sequence number is specified, the statement can be the object of a branch. Internally, the COMPUTE statement is processed as shown:

```
028   REVENUE  -  COST      WORK*01
028   WORK*01  /  COUNT     AVE-PROFIT
```

The difference in total revenue and total cost is divided by COUNT, which signifies the number of records processed by Report 02. The result of this computation is stored in the work field, AVE-PROFIT. The total values of REVENUE, COST, and COUNT and the calculated value of AVE-PROFIT can be output on total lines each time a control break executes and at the end of the output phase.

Example 3: Conditional Operation—Simple

```
227      CRED-LIMIT GT 500.00 TAKE
```

When the value of CRED-LIMIT is greater than $500.00 dollars, any remaining type 7 procedure logic for Report 22 is bypassed and the current buffer is extracted. If the comparison is false, the next type 7 process statement is processed.

An alternative method for coding this procedure logic appears below:

```
22I   CRED-LIMIT LE 500.00 DROP
22I   TAKE
```

Example 4: Conditional Operation—Compound

```
 467030 IF DAY.INDX NE ('MON' 'WED' 'FRI')
*       OR INIT EQ ('A' TO 'L') DROP
```

Two comparisons are specified in this compound test statement. An occurrence of the multiply-occurring field DAY is compared to a list of alphanumeric literals for a NE (not equal) condition. INIT is compared to a range of alphanumeric values for an equal condition. If one of these tests is true, DROP executes; that is, a DROP is performed when either the value of DAY.INDX does not equal MON, WED, or FRI or the value of INIT is in the range 'A' through 'L', inclusive. An input buffer that is not dropped continues processing in the 7 logic of Report 46.

Example 5: Conditional Operation—EOF

```
8951*010 TOTAL-BAL
897010  EOF EQ TAKE
897020  COMPUTE CURR-BAL + TOTAL-BAL  TOTAL-BAL
897030  DROP
```

An end-of-file condition is tested for each record processed. When end-of-file is reached, the type 7 procedure logic is entered one more time. Since the first type 7 statement specifies a TAKE action, any remaining type 7 process statements are not executed. If the EOF test were coded after the COMPUTE and DROP statements, the detail line for the report would not be extracted since the DROP action would occur before the EOF statement.

The code in this example achieves the same result as a totals-only report that contains no procedure logic, as shown below:

```
89       OUT T
8951*010  CURR-BAL    $CURR-BAL REPLACES TOTAL-BAL
```

Example 6: Conditional Operation—EOF

Sample code containing multiple EOF tests follows:

```
027    IF EOF EQ 200
 .
 .          ◄- Processing logic for input records
 .
027200      ◄- Processing logic for input records and EOF
 .
 .
 .
027    IF EOF EQ DROP
 .
 .          ◄- Processing logic for input records
 .
```

When the end of the file is reached, the type 7 code is entered one last time. The first type 7 statement passes processing control to the process statement with sequence number 200. The statements coded between sequence number 200 and the second EOF test execute.

Example 7: Conditional Operation—LEVL

```
01SORT DIVISION,-,DEPARTMENT,0,TITLE,EMP-LNAME
128    IF LEVL EQ 2 100
128    IF LEVL EQ 3 200
128    MOVE 'DEPARTMENT TOTALS' TO LABEL
128    TAKE
128100 MOVE 'DIVISION TOTALS' TO LABEL
128    TAKE
128200 MOVE 'COMPANY TOTALS' TO LABEL
```

This type 8 process parameter tests the level of the control break during break processing. Level 1 refers to the control break associated with DEPARTMENT on the SORT parameter; level 2 refers to the control break associated with DIVISION. The value of LEVL at grand totals time is one more than the number of control breaks specified on the SORT parameter. In this example, the value of LEVL is 3.

Depending on the level of the control break, the literal DEPARTMENT TOTALS, DIVISION TOTALS, or COMPANY TOTALS moves to LABEL. When TAKE executes, the total lines are output and processing for the current control break stops. A TAKE automatically executes after the last procedure statement, so an explicit statement is not coded after sequence number 200.

Example 8: Conditional Operation—Branch

```
627     B 300
```

Processing control passes unconditionally to the type 7 process parameter with a sequence number of 300.

Example 9: Conditional Operation—Branch

```
2541*001 ACCOUNT-NUM
257010   B HEAD
```

In this example, ACCOUNT-NUM is referenced on a type 4 edit parameter, but not on a SORT parameter for Report 25. Therefore, it is necessary to code a B HEAD statement so that values of ACCOUNT-NUM can appear as report headers.

Example 10: Assignment Operation—MOVE

```
437     MOVE 'AMT NOW DUE' TO DISPLAY-MESSAGE
```

In this example of a MOVE operation, the literal 'AMT NOW DUE' is moved to a work field, DISPLAY-MESSAGE. If the length of DISPLAY-MESSAGE is smaller than the length of the literal, the excess characters in the literal truncate on the right; when this happens, CA Culprit issues a W-level message in the Input Parameter Listing.

In the example presented below, DISPLAY-MESSAGE is set to blanks, by coding a single blank in the MOVE operation. Since the field being moved is smaller than the result field, the rightmost characters of DISPLAY-MESSAGE are padded with blanks.

```
437     MOVE ' ' TO DISPLAY-MESSAGE
```

Example 11: Assignment Operation—MOVE

```
548010 IF LEVL NE 2 070
548020 MOVE 0 TO PAGE
```

The system-maintained work field PAGE is reset to 0 when the current control break is 2. Because the value of PAGE increases by 1 before printing, the next page appears in the title as PAGE 1. When the control break does not equal 2, processing control passes to the type 8 process parameter that has a sequence number of 070.

Example 12: Assignment Operation—MOVE

```
0151*020 PRINT-NUM
017      MOVE NUMERIC-WORK-FLD.2 TO PRINT-NUM
```

The second occurrence of a multiply-occurring work field, NUMERIC-WORK-FLD, is moved to another work field, PRINT-NUM, for output during input processing. To perform this operation, PRINT-NUM must be defined as numeric.

Example 13: Assignment Operation—CONVERT

```
670  WORK-FIELD
677     CONVERT '     128' TO WORK-FIELD
```

The alphanumeric literal (space)(space)(space)(space)(space)128 is converted to the numeric value 00000128 (space represents blanks); this value is placed in a numeric work field, WORK-FIELD.

Example 14: Assignment Operation—CONVERT

```
020 WORK-DP2  DP=2  ALPHA-WORK '12345.67'
027    CONVERT ALPHA-WORK TO WORK-DP2
```

In this example, the alphanumeric and numeric work fields both contain two decimal positions. The CONVERT statement assigns WORK-DP2 the numeric equivalent of the alphanumeric literal.

If ALPHA-WORK had a value with three decimal positions, for example: 12345.678, the numeric value assigned to WORK-DP2 would be rounded to two decimal positions: 12345.68.

Example 15: Control Operation—PERFORM/RETURN

```
337010 IF BALANCE LE 200 860
337020 PERFORM 810
         -                  ◄- More procedure logic
         -
337    TAKE
337810 BALANCE + TOT-BAL TOT-BAL
337    COUNT + 1 COUNT
337    RETURN
337860 MOVE LOW-BAL-MESSAGE TO MESSAGE
```

In the sample code above, processing control is passed to the process parameter that has sequence number 810 when the input buffer value for BALANCE is greater than 200. TOT-BAL and COUNT are incremented, and processing control returns to the process statement that follows the PERFORM statement. A TAKE instruction selects the record for printing and bypasses remaining process code.

If BALANCE is less than or equal to 200, a message is transferred to a work field referenced on a type 5 edit parameter. An implicit TAKE at the end of type 7 logic selects the record for printing.

Example 16: Control Operation—CALL

```
997    CALL US11 (JUL-DATE, GREG-DATE)
```

This CALL statement invokes a CA-supplied procedure module, CULLUS11 (Julian Date Conversion). (See *CA Culprit for CA IDMS User Modules Guide* for details on coding this procedure module.) Two arguments are passed to the procedure module: JUL-DATE corresponds to ARG1; GREG-DATE corresponds to ARG2.

Internally, CA Culprit processes the CALL statement shown above as follows:

```
997   MOVE  JUL-DATE  ARG1
997   MOVE  GREG-DATE ARG2
997   B     US11
```

Example 17: Control Operation—PICK/TAKE

```
6651*010 'BALANCE'
6651*020 POSITIVE-BALANCE
6651*030 ACCOUNT-NUM
6652*010 'NEGATIVE BALANCE'
6652*020 NEGATIVE-BALANCE
6652*030 ACCOUNT-NUM
6653*030 NUM-TRANSACTIONS
667    PICK 3
667    IF AMOUNT LT 0  040
667    MOVE AMOUNT TO POSITIVE-BALANCE
667    TAKE 1
667040 MOVE AMOUNT TO NEGATIVE-BALANCE
667    TAKE 2
```

In this example, detail line 3 is always selected to be extracted, as specified by the PICK statement. It is extracted, however, as a result of a subsequent TAKE statement. Detail line 1 is extracted when the value of AMOUNT is greater than or equal to 0; detail line 2 is extracted when the value of AMOUNT is less than 0.

Example 18: Control Operation—TAKE

```
2261*0100'DIVISION TOTAL'
2261*040 TOTAL-SALARIES
2262*010 'NUMBER OF EMPLOYEES'
2262*040 COUNT
2263*040 ANNUAL-EMP-SALARY
2264*040 STATUS
228    IF LEVL NE 1  070
228    TAKE (3 4)
228070 TAKE (1 2)
```

Edit lines 1 and 2 are selected when the control break is not 1; edit lines 3 and 4 are selected when the control break is 1. When a new control break executes, the status of all type 6 edit parameters is set to deselect because line selection specifications are coded on type 8 process parameters.

Example 19: Control Operation—RELS/TAKE

```
017    COMPUTE BAL-DUE  -  PAYMENT  BAL-DUE
017    RELS (1,2,4)
017    IF BAL-DUE GT 0 DROP
017    TAKE 3      $THIS LINE FLAGS ZERO/NEGATIVE BALANCES
```

After the arithmetic process statement executes, edit lines 1, 2, and 4 are extracted to the extracted items and statistics file. If a positive balance remains due, processing ends for this record. If a negative or zero balance remains due, edit line 3 is extracted also.

Internally, the RELS statement is processed as shown below:

```
017    PICK (1 2 4)     $SELECT LINES 1 2 and 4
017    RELS             $PRINT LINES 1 2 and 4
017    UNPICK (1 2 4)   $DESELECT LINES 1 2 and 4
```

The RELS statement resets the select/deselect status of edit lines 1, 2, and 4; edit line 3 must be explicitly selected in the TAKE statement in order to print. When a RELS immediately follows a PICK, the user must explicitly deselect lines coded on the PICK instruction, as appropriate.

Example 20: Control Operation— DROP

```
347010 IF SEX EQ 'F' DROP
347020 IF AGE LE 65  DROP
    -
    -                    ◄-  More procedure logic
```

Records of all males over the age of 65 are selected for processing in report 34. All other records are dropped.

Chapter 6: Work Field Parameters

This section contains the following topics:

About Work Fields Parameters

Work field parameters define work fields for use during report and SELECT/BYPASS processing. Work fields are either report specific or global. A global work field is available to all reports in the run; a report-specific work field is available only to the report for which it is defined.

Uses

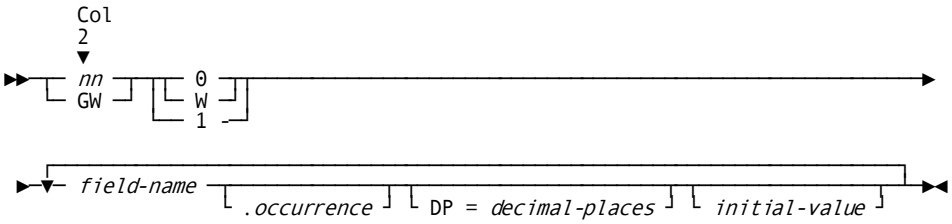
Work fields can be defined as numeric or alphanumeric. Numeric work fields are commonly used as either counters or as result fields for arithmetic calculations coded on type 7 and type 8 process parameters. Alphanumeric work fields are commonly used to transfer error messages and labels to output fields coded on type 4, 5, or 6 edit parameters.

Work Field Parameters

Purpose

Define work fields for use during report and SELECT/BYPASS processing.

Syntax



Syntax Rules

nn

Identifies the report with which the work field is associated. It is a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

GW

Identifies a global work field that is available to all reports in the run. GW must be coded starting in column 2.

The value of a global work field passed to a report depends on whether the field value is altered by the type 7 or type 8 procedure logic in the previous report. The value is determined as follows:

- **Type 7** procedure logic executes for each record read from the input buffer. The record passes through type 7 procedure logic for all reports in the run before another record is read. The value of the global work field at the end of type 7 procedure logic in one report is passed to the type 7 procedure logic of the next report with each record processed.

  The value of the global work field at the end of type 7 procedure logic for the highest numbered report in the run is passed to the type 7 procedure logic of the first report of the run when the next input record is processed.

- **Type 8** procedure logic executes each time a control break executes and at the end of the output phase. The value of a global work field at the end of the output phase of one report is passed to the output phase of the next report.

The following table illustrates how the value of a global work field can change from report to report in a CA Culprit run. Global work fields are commonly used as a constant value accessed by more than one report in a run. Variable global work fields are commonly used to accumulate totals for an entire CA Culprit run; for example, all records processed for all reports can be counted.

Global Work Field Processing

Logic	Process Statement	Buffer Number	Global Value
7			0
7	017 GLOBAL + 1 GLOBAL	1	1
	027 GLOBAL TIMES 3 GLOBAL	1	3
	017 GLOBAL + 1 GLOBAL	2	4
	027 GLOBAL TIMES 3 GLOBAL	2	12
8	018 GLOBAL DIVIDE 6 GLOBAL		2
	028 GLOBAL MINUS 2 GLOBAL		0

Note: This table illustrates the value of a global work field during type 7 and type 8 procedure logic; the only control break occurs at the end of the output phase. In this example, two input buffers are processed in reports 01 and 02. The global work field is type 0 (GW0) and is initialized to 0.

0

Specifies that the value of the work field is initialized only at the start of the extract phase. The value of the work field at the start of the output phase depends on whether the field is defined as numeric or alphanumeric, and whether the field is referenced on a SORT parameter, a type 5 edit parameter, or both.

Must be coded in column 4.

1

Specifies that the value of the work field is initialized at the start of the extract phase and also at the start of the output phase under the following circumstances:

- If the work field is numeric and is not referenced on a SORT parameter or a type 5 edit parameter
- If the work field is alphanumeric and is not referenced on a SORT parameter

In all other cases, the value of a type 1 work field at the start of the output phase is the same as the value of a type 0 work field.

Must be coded in column 4.

field-name

A 1- to 32-character name that identifies the work field. It can be coded in any column after 4. The name can consist of letters, numbers, or hyphens. The following rules apply to specifying *field-name*:

- At least one alphabetic character is required.
- The name cannot begin or end with a hyphen.
- The name cannot be any of the reserved words listed in Reserved Words (see page 369).

occurrence

The number of occurrences of the work field. It is specified only if the work field occurs more than once. *Occurrence* can be any number in the range 0 through 32767; if specified, *occurrence* must be separated from *field-name* by a period (.).

In the extract phase, the subscript always acts as a pointer to a particular occurrence of a multiply-occurring field. In the output phase, the subscript is considered part of the field name if the work field is specified on a SORT or type 5 edit parameter; otherwise, the subscript is interpreted as a pointer to an occurrence of work field values.

A value of zero for *occurrence* has a special meaning when it is associated with an alphanumeric work field (see *initial-value* below). A subscript of zero indicates that the work field redefines the storage area that immediately follows it.

The work field definitions in the redefined area must be coded so they occur immediately after the zero-subscripted work field when CA Culprit completes the parameter sort phase of processing. The work field parameters are sorted first by work field type (0 or 1), then by data type (alphanumeric or numeric), then by field size if the field is numeric (8-byte or 16-byte), and then by field name before processing begins.

The subscript value of a zero-subscripted work field referenced on a SELECT/BYPASS, SORT, edit, or process parameter must be a positive integer, for example, WORK-FIELD.2. The subscript value multiplied by the length of a single occurrence of the zero-subscripted work field should not exceed the length of the redefined area. The subscript value can be tested in procedure logic.

Note: For more information on zero-subscripting, see the *CA Culprit for CA IDMS User Guide*.

DP = ***decimal-places***

Specifies the number of digits to the right of the decimal point in a numeric work field. This value must be in the range 0 through 31. If DP=*decimal-places* is not coded and the initial work field value does not specify decimal places (see *initial-value* below), the work field is treated as an integer in all calculations used in selection logic, procedure logic, or automatic totaling.

When DP= is specified, the number of digits to the right of the decimal point in the initial value of any singly- or multiply-occurring work field must not exceed the number coded for *decimal-places*. Rules regarding the position of a decimal point when printing a work field appear in Output Definition Parameters (see page 69).

Note: When the PROFILE parameter option RELEASE=5 is specified and DP= is not specified, the default is 2.

initial-value

This is the value associated with the named work field when the field is established and, in the case of type 1 work field parameters, when the field is reinitialized at the start of the output phase.

If *initial-value* is not specified, an initial value of 0 is assigned to the work field. If the work field occurs multiple times, an initial value can be specified once for each occurrence. The first value applies to the first occurrence, the second value applies to the second occurrence, and so forth. If fewer values than occurrences are specified, the remaining occurrences are set to the last initial value specified.

Initial-value implicitly describes the field length and data type for alphanumeric, hexadecimal, and numeric values, as follows:

- An **alphanumeric value** is a 1- to 64-character alphanumeric literal enclosed in single quotation marks. To express an apostrophe (a single quotation mark), code two consecutive single quotation marks; for example, 'ERNIE''S DINER'.

  If the field occurs multiple times, the length of all occurrences is the same and is determined by the length of the first initial value specified. Subsequent values are padded with spaces or truncated on the right, as necessary.

- A **hexadecimal value** is a 1- to 50-character (25-byte) hexadecimal literal, preceded by X and enclosed in single quotation marks; for example, X'0A2C'.
- A **numeric value** is a 1- to 31-digit numeric literal that can contain a leading sign and a trailing or embedded decimal point; for example, 12.34. A leading number must precede the decimal point; for example, 0.2. Commas and other editing characters are not permitted. If no sign is specified, the value is treated as positive.

  Numeric work fields are stored internally as an 8-byte signed packed decimal unless one of the following conditions exists:

  – The initial value contains more than 15 digits.

  – A DP= specification is coded.

  – The initial value contains a decimal point.

  In these cases, the work field is stored internally as a 16-byte signed packed decimal.

The number of decimal places in a multiply-occurring work field is determined by the DP= specification, if coded, or by the number of decimal positions specified in the first initial value. If the number of decimal positions in subsequent initial values exceeds the number specified for the first value, an E-level error results.

Examples

Sample work field parameters are shown and described below.

Example 1

```
010  COUNTER 1

0151*000 COUNTER
0161*010 COUNTER
```

Report 01 contains a numeric work field COUNTER that is initialized to 1 at the start of the extract phase. The parameter type (0) indicates that COUNTER is not reinitialized at the start of the output phase. If COUNTER were a type 1 work field that was not referenced on a SORT parameter or on a type 5 edit parameter, the value of COUNTER would be reset to one at the start of the output phase.

In this example, COUNTER is referenced as a nonprinted output field on a type 5 edit parameter. The total value of COUNTER is output on a total line.

Example 2

```
351  MESSAGE-AREA  '               '
```

MESSAGE-AREA is a 15-character alphanumeric work field; the length is determined by counting the number of spaces between the quotation marks.

This work field can be used to display unique messages by moving literals into the work field in procedure logic and referencing the work field on an edit parameter. At the start of the output phase, MESSAGE-AREA is reinitialized to blanks if the field is not a sort-key value; otherwise, MESSAGE-AREA contains the value of the sort key at the control break.

Example 3

```
240  COUNT LITERAL.3,'A','B','C' FIELDC 'XXX',TRIGGER.2 DP=2 0 1
```

The following work fields are established in this example:

- COUNT is a numeric 8-byte packed field; it is initialized to zero.
- LITERAL is a 1-character alphanumeric field that occurs three times; initial values are A for the first occurrence, B for the second occurrence, and C for the third occurrence. Each occurrence is referenced on SELECT/BYPASS, SORT, edit, and process parameters by a subscript value; for example, the second occurrence of LITERAL can be referenced as LITERAL.2 or LITERAL.INDX, where INDX has a value of 2.
- FIELDC is a 3-character alphanumeric field initialized to XXX.
- TRIGGER is a 16-byte numeric field that occurs two times; initial values are 0.00 for the first occurrence and 1.00 for the second occurrence.

Example 4

```
GW0  TABLE.20  DP=4
```

TABLE contains 20 occurrences, each initialized to 0.0000. Each occurrence of TABLE is a 16-byte packed signed decimal. CA Culprit reserves 320 bytes of storage to accommodate this field. Values for each of these fields can be set and referenced by any report in the CA Culprit run; for example, the seventh occurrence can be referenced as TABLE.7 (or, for example, TABLE.COUNT where COUNT is assigned a value of 7) on SELECT/BYPASS, SORT, edit, and process parameters defined for all reports.

Example 5

```
150  HEX.16  X'0A'  X'0A'  X'1E'  X'14'  X'14'  X'00'
```

Report 15 contains 16 hexadecimal literals in the work field HEX. The first six occurrences of HEX are explicitly initialized. The remaining 10 occurrences are initialized to X'00', which is the value of the last occurrence specified.

Hexadecimal fields are treated as though they were alphanumeric, so no arithmetic manipulations can be performed on this work field. The hexadecimal work fields are stored internally as two hexadecimal digits for each byte; therefore, each occurrence of HEX uses one byte of storage.

Example 6

```
270  ALPHA.10  'ONE    ' 'FIVE' 'TEN' 'ELEVEN' 'THIRT''N' ' '
```

This report contains ten 7-character alphanumeric work fields; the first initial value establishes the length of all ten occurrences. The first six occurrences are initialized as shown; the last four occurrences are set to spaces, which is the value of the last literal specified. To express an apostrophe in the fifth value, two consecutive single quotation marks are coded; these are stored as a single quotation mark.

Example 7

```
020  POINT.5   DP=3  3.6  -9.141   5.000
```

POINT contains five occurrences, each containing three decimal positions. The following initial values are assigned: 3.600, -9.141, 5.000, 5.000, and 5.000. Each occurrence in POINT is a 16-byte packed signed decimal; 80 bytes of storage are required to accommodate this field.

Example 8

```
010  AMT.5   SUBS

0151*010 AMT.SUBS
0161*010 AMT.SUBS
0161*020 AMT.3
```

During the extract phase, every occurrence of AMT.SUBS is extracted, as indicated by the current value of SUBS. During the output phase, every extracted value of AMT.SUBS is summed, regardless of the value of SUBS during the extract phase. SUBS is the name of a total field during the output phase and no longer has a unique value as a subscript.

The value of AMT.3 during the output phase is the current value of the third occurrence of the work field, provided AMT.3 is not specified on a SORT parameter or on a type 5 edit parameter.

Example 9

```
REC START-YEAR  70  2    'START YEAR'
REC START-MONTH 72  2    'START-MONTH'

140 START-DATE-0.0'     '       $START-DATE IS 5 BYTES LONG
140 START-DATE-1  '  '          $MONTH
140 START-DATE-2  '/'
140 START-DATE-3  '  '          $YEAR

147010 MOVE START-YEAR TO START-DATE-3
147010 MOVE START-MONTH TO START-DATE-1
1451*010  START-DATE-0.1        $PRINT MM/YY
```

START-DATE-0.0 defines a zero-subscripted alphanumeric work field and redefines the storage area that immediately follows with 5-byte occurrences. The work fields in the redefined area are coded in alphabetical order (that is, -1, -2, and -3) so that the fields sort immediately after START-DATE-0 in the parameter sort phase of CA Culprit processing.

CA Culprit reads values for START-YEAR and START-MONTH from the input file and moves the values to the redefined work field storage area. The date in *mm/yy* format, is output each time the type 5 edit parameter is processed. START-DATE-0.1 specifies the first occurrence of 5 bytes in the redefined work field area; that is, the 5 bytes that start with START-DATE-1.

Example 10

```
020 DATE-A.0 '  '           $DATE-A IS 2 BYTES LONG
020 DATE-B   'MMYY'
0251  10 DATE-A.2           $PRINT YY
0251  12 '/'                $PRINT /
0251  13 DATE-A.1           $PRINT MM
```

Work field DATE-A.0 is initialized to 2 blanks. DATE-B is initialized to a 4-byte literal value. The type 5 edit parameters are coded so that the second occurrence of 2 bytes in DATE-B (YY) prints before the first occurrence of 2 bytes (MM). The output field will appear as YY/MM.

More information:

Process Parameters (see page 103)
Introduction (see page 13)
Output Phase Field References (see page 309)
Release 5 and 6 Default Actions (see page 365)

Chapter 7: Implied Subscript Parameters

This section contains the following topics:

Uses

The implied subscript parameter provides an alternative name for a multiply-occurring input or work field. This parameter is commonly used to simplify references to specific occurrences of multiply-occurring fields. It also provides an optional method of outputting tables of information.

A reference to a field defined on an implied subscript parameter is a reference to the multiply-occurring field and to the index value associated with that field on the implied subscript parameter.

Example

An example appears below:

```
4415 LOCATION LOC-NUM.COUNT
```

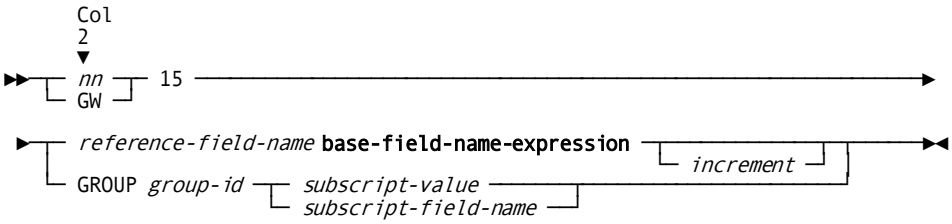
In this example, LOC-NUM.COUNT identifies a specific occurrence of the multiply-occurring field LOC-NUM. The implied subscript parameter renames the field LOCATION. A reference to LOCATION on a SELECT/BYPASS, SORT, edit, or process parameter implies a reference to a specific occurrence of LOC-NUM, depending on the value of COUNT.

Implied Subscript Parameters

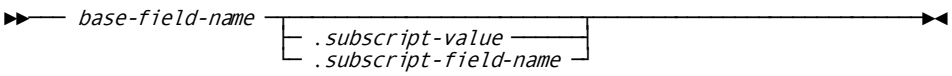
Purpose

Provides an alternative name for a multiply-occurring input or work field.

Syntax



Expansion of Base-field-name-expression



Syntax Rules

nn

Identifies the report with which the implied subscript field is associated. *Nn* must be a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

GW

Indicates that the implied subscript work field is available to all reports in the run. GW must be coded starting in column 2.

15

Identifies the parameter as an implied subscript definition. The value must be coded in columns 4 and 5. Column 6 must remain blank.

reference-field-name

Is the alternative name for a specific occurrence of a multiply-occurring field. The name can consist of letters, numbers, or hyphens. The following rules apply to specifying *reference-field-name*:

- At least one alphabetic character is required.
- The name cannot begin or end with a hyphen.
- The name cannot be any of the reserved words listed in Reserved Words (see page 369).

base-field-name-expression

Is the name of the multiply-occurring field. See expansion of *base-field-name-expression* below:

base-field-name

Specifies the name of a multiply-occurring input field defined on a REC parameter that specifies the keyword ELMNT, or a multiply-occurring work field defined on a work field definition parameter.

subscript-value

Is a numeric literal that identifies a specific occurrence of *base-field-name*; for example, EMPLOYEE-NUM.2 specifies the second occurrence of EMPLOYEE-NUM.

subscript-field-name

Is the name of a singly-occurring numeric input field or work field that identifies an occurrence of *base-field-name*; for example, if INDX has a value of 2, EMPLOYEE-NUM.INDX specifies the second occurrence of EMPLOYEE-NUM. The subscript field definition must not specify a decimal point.

Subscript-value and *subscript-field-name* must be separated from *base-field-name* by a period (.). The value of the subscript should be in the range 1 to *n*, where *n* is the number of repetitions of the field name. The value of the subscript can be tested in procedure logic.

CA Culprit treats a reference to *reference-field-name* as though it were a reference to *base-field-name-expression*, as illustrated in the following example:

```
13510001 EMP-NUM
1315 EMP-NUM  EMPLOYEE-NUM.INDX
```

In the above example, a reference to EMP-NUM on a type 5 edit parameter implies a reference to a specific occurrence of EMPLOYEE-NUM depending on the value on INDX.

increment

Is a numeric literal that is added to the subscript value before an occurrence of *base-field-name* is referenced. *Increment* must be in the range 1 through 4095; the total of the increment plus the subscript value must not exceed the number of occurrences of *base-field-name*.

The value of the subscript remains unchanged; therefore, one occurrence of the multiply-occurring field can be referenced relative to another occurrence of the same field.

GROUP

Establishes a common subscripting field for a group of multiply-occurring input fields.

An implied subscript parameter that specifies the keyword GROUP replaces a reference to an alternative field name defined on an implied subscript parameter that specifies *reference-field-name*. If the GROUP implied subscript option is specified, members of a group of multiply-occurring fields must not be referenced with explicit subscripts for the report for which the implied subscript parameter is defined. GROUP identifies an implied subscript parameter that establishes an index for all members of a group of multiply-occurring input fields. Any reference to an element within the group is interpreted at run time as a reference to the subscripted element. Only one type 15 parameter may be coded for each REC parameter that specifies the keyword GROUP in each report.

group-id

Identifies the associated group, as coded on a REC parameter that specifies GROUP. *Group-id* must be a 2-byte alphanumeric literal that must be unique for the run.

subscript-value

Is a numeric literal that is the explicit subscript associated with the group.

subscript-field-name

Is the name of a singly-occurring numeric work field whose value is the explicit subscript associated with the group. The subscript field definition must not specify a decimal point.

The value of the subscript should be in the range 1 to *n*, where *n* is the number of group repetitions. The subscript value can be tested in procedure logic.

CA Culprit treats a reference to an element within the group as though it were a reference to the field name subscripted by *subscript-value* or *subscript-field-name*, as shown in the example below:

```
REC EMP-INFO 20  GROUP  AA 10.5
REC EMP-ID   1 3 ELMNT  AA
13510010 EMP-ID
1315 GROUP AA 3
```

In the above example, EMP-ID is a multiply-occurring input field associated with GROUP AA; EMP-ID occurs five times. A reference to EMP-ID on a type 5 edit parameter implies the third occurrence of this field as determined by the subscript value defined on the type 15 parameter.

Examples

Sample implied subscript parameters are shown and described below.

Example 1

```
REC CUST-INFO        35      GROUP CC  40.6
REC CUSTOMER-NAME     1  20  ELMNT CC  'CUSTOMER NAME'
REC CUSTOMER-ADDRESS 21  20  ELMNT CC  'CUSTOMER ADDRESS'

0115  NAME  CUSTOMER-NAME.3
```

NAME is defined as the third occurrence of CUSTOMER-NAME. CUSTOMER-NAME is an element of GROUP CC specified on a REC parameter; it occurs six times.

Example 2

```
010   INDX  1
0115  NAME  CUSTOMER-NAME.INDX
```

CUSTOMER-NAME is an element of GROUP CC defined in the preceding example. NAME is defined as an occurrence of CUSTOMER-NAME, which varies depending on the value of INDX; INDX is a numeric work field initialized to 1.

Example 3

```
230  AMT.3   1 2 3   SUBS
2315 TOTAL1  AMT.1
2315 TOTAL2  AMT.2

2351*010 AMT.SUBS
2361*010 TOTAL1
2362*010 TOTAL2
```

TOTAL1 and TOTAL2 are alternate names for the first and second occurrences of the work field AMT; in this example, TOTAL1 is assigned an initial value of 1 and TOTAL2 is assigned a value of 2. AMT.SUBS accumulates all specific occurrences of AMT in the extract phase. The total value of AMT.SUBS can be referenced during the output phase. References to TOTAL1 and TOTAL2 during the output phase produce the current values of the first and second occurrences of the array that stores values of AMT.

Example 4

```
REC LOCATION-INFO    20     GROUP AA   8.5
REC LOCATION-ID       1  2  ELMNT AA
REC EFFECTIVE-DATE    3  6  ELMNT AA
01520001 EFFECTIVE-DATE

010   INDX 3
0115  GROUP AA INDX
```

INDX is the subscript for the elementary fields within GROUP AA; it is a work field initialized to 3. The reference to EFFECTIVE-DATE on a type 5 edit parameter implies the third occurrence of this field. If the value of INDX changes, references to LOCATION-ID and EFFECTIVE-DATE imply the field occurrences equal to the value of INDX. The value of INDX cannot exceed 5, the number of occurrences of GROUP AA. Because an implied subscript is coded for GROUP AA, explicit references to the elements of GROUP AA are not permitted for report 01; for example, the following reference would be invalid:

```
01610010 LOCATION-ID.4
```

Example 5

```
 220 IX 1    TABLE.28 'A' 'B' 'C' 'D' 'E' 'F' 'G'
*                    'H' 'I' 'J' 'K' 'L' 'M' 'N'
*                    'O' 'P' 'Q' 'R' 'S' 'T' 'U'
*                    'V' 'W' 'X' 'Y' 'Z'' '
 2215 COLUMN1  TABLE.IX
 2215 COLUMN2  TABLE.IX  1     $INCREMENT IX BY 1
 2215 COLUMN3  TABLE.IX  2     $INCREMENT IX BY 2
 2215 COLUMN4  TABLE.IX  3     $INCREMENT IX BY 3
 227010 PERFORM 500            $CODE EVALUATING VALUE OF IX
 227020 TAKE
 227500 RELS
 227550 IX ADD 4 IX
 227560 IF IX LE 24 500
 227570 RETURN
 2251*010 COLUMN1
 2251*020 COLUMN2
 2251*030 COLUMN3
 2251*040 COLUMN4
```

In this example, TABLE is a multiply-occurring work field with 28 entries; IX is a numeric work field initialized to 1. COLUMN1, COLUMN2, COLUMN3, and COLUMN4 are alternative names for subscripted values of TABLE; these are defined as implied subscript parameters. An increment is used on all but the first implied subscript parameter for the purpose of automatically incrementing the subscript value of TABLE.

In type 7 procedure logic, the value of IX increases by 4 until it is greater than 24. Before each increment, CA Culprit extracts the type 5 edit parameter information.

An example of the report output appears below:

```
A           B           C           D
E           F           G           H
I           J           K           L
M           N           O           P
Q           R           S           T
U           V           W           X
Y           Z
  RECORDS WRITTEN FOR REPORT 22—        7
```

More information:

Input Definition Parameters (see page 43)

Chapter 8: Copied Code Parameters

This section contains the following topics:

Advantages

Frequently used portions of CA Culprit code can be stored and accessed by several reports or users. By using stored code, the number of parameters input at run time is reduced, and standard file definitions and procedures are established. For example, users can store INPUT and REC parameters that define a shared input file. Users can also store type 7 and type 8 process parameters that perform an operation common to several reports.

Stored Code

Stored code, which can be inserted in a parameter stream at run time, can be maintained in card decks, the data dictionary, partitioned data sets in a z/OS environment, source statement libraries in a z/VSE environment, CA Panvalet libraries, or CA Librarian libraries. At a z/VM installation, the source can be retrieved from a z/VM MACLIB or a z/OS partitioned data set. The stored code is inserted during the precompile phase of CA Culprit processing.

Three parameters are available to copy and modify stored code, as follows:

- **USE**
- **=COPY**
- **=MACRO**

Inline code can also be modified by using these parameters. Although the copied or inline code is modified for a particular CA Culprit run, the modifications do not alter the code that is stored.

Capabilities of Copied Code Parameters

The following table highlights the capabilities of each copied code parameter. Each parameter is discussed separately below, followed by a discussion of the =MACRO AMLIST routine, which can be used to list the fields of input records.

Function	=COPY	=MACRO	USE
Copies stored code	●	●	●
Substitutes arguments for symbolic parameters		●	●
Assigns default values to symbolic parameters			●
Parameter nesting			●
Parameter changes			
Report number	●	●	●
Process sequence numbers		●	●
Edit parameter type		●	●
Edit line number		●	●
Edit column number		●	●
Character string replacement			●
Parameter suppression			
INPUT	●	●	●
REC		●	●
SELECT/BYPASS			●
OUTPUT		●	●
SORT		●	●
Title		●	●
Edit		●	●

Function	=COPY	=MACRO	USE
Process		●	●
Work field		●	●
By character string			●
By identification code			●

More information:

Introduction (see page 13)

USE Parameter Overview

The USE parameter provides all the capabilities of the =COPY and =MACRO parameters, as well as special capabilities that are unique to USE. The amount of coding required to process stored code is reduced with the USE parameter compared with =COPY and =MACRO parameters, and the number of tasks handled internally is increased.

Capabilities that are unique to the USE parameter appear below:

- **Keywords and key values** that correspond to symbolic parameters can be coded; these expressions can be used to assign values to the symbolic parameters. Symbolic parameters appear in copied code in the form &&1 through &&9 and &&A through &&Z.
- **Default values** can be defined for symbolic parameters. If no user-coded value is provided for the symbolic parameter, the default value is used during processing.
- **Null values** can be defined in character strings that can be used as values for symbolic parameters.
- **Character strings** can be replaced.
- **CA Culprit parameters** can be eliminated based on the processing requirements of a particular report.
- **USE parameters** can be nested; that is, the USE parameter can appear in the code that is copied.

The following table provides an overview of the parameters and clauses associated with the USE parameter.

Clause	Tasks Performed
WITH VALUES	Defines values for symbolic parameters

Clause	Tasks Performed
CHANGE	Requests changes in copied or inline code
DROP/KEEP	Drops or keeps parameters based on individual processing needs
RENUMBER	Resequences type 7 and type 8 process parameters
DEFAULT	Associates keyword expressions with symbolic parameters and defines default values for these parameters
END	Indicates the end of a portion of inline CA Culprit code processed by using a USE * parameter

One or more USE parameters can be nested in copied code. Each USE parameter in the copied code is added to the nesting hierarchy; the hierarchy determines the number of levels of CA Culprit processing. There is no limit to the number of nesting levels.

The following two figures illustrate the CA Culprit input parameters that include nested USE parameters and the Sequential Parameter Listing for these input parameters respectively. CA Culprit modifies the code and outputs the changes on the Sequential Parameter Listing. CA Culprit also outputs a two-digit number that indicates the level of nesting and the source of the modified code; in this example, SYSIN indicates that the source of code is inline. The nesting information is output following each USE statement and, in this example, following each END clause.

Inline Code Containing Nested USE Parameters

```
USE * CHANGE 'TITLE' TO 'PRINCE'
 IN  80 DD=EMPFILE
 REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
 REC EMP-LNAME  11 15
 REC SALARY     70 10 3 DP=2
 REC TITLE      50 20         'TITLE'
USE *
CHANGE 'AVG-SAL' TO 'AVERAGE-SAL' AND R TO 22
 010 COUNT 1
 010 AVG-SAL DP=2 0
USE * CHANGE ',0,' TO ',-,'
 013EMPLOYEE SALARY REPORT
 01SORT TITLE,0,EMP-LNAME
END
 01510000 COUNT
 0151*010 TITLE HR   SZ=20
 0151*020 EMP-NAME HR
 0151*030 SALARY F$ SZ=10 HF
USE * DROP '62'
 0161*030 SALARY SZ=10 F$
 0162*010 LABEL.2
 0162*030 COUNT SZ=10
END
USE * RENUMBER 8
 018500   SALARY / COUNT  AVG-SAL
END
END
END
```

Sequential Parameter Listing of Code Modified by Nested

```
mm/dd/yy                    SEQUENTIAL PARAMETER LISTING    Vnn.n  PAGE     1

00 ** SYSIN **                          USE * CHANGE 'TITLE' TO 'PRINCE'
01 ** SYSIN **                           IN  80 DD=EMPFILE
  INSTALLATION SECURITY OPTION IS NO
                                         REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
                                         REC EMP-LNAME  11 15
                                         REC SALARY     70 10 3 DP=2
                                         REC PRINCE      50 20         'PRINCE'
                                        USE *
                                        CHANGE 'AVG-SAL' TO 'AVERAGE-SAL' AND R TO 22
02 ** SYSIN **                           220 COUNT 1
                                         220 AVERAGE-SAL DP=2 0
                                        USE * CHANGE ',0,' TO ',-,'
03 ** SYSIN **                           223EMPLOYEE SALARY REPORT
                                         22SORT PRINCE,-,EMP-LNAME
                                        END
02 ** SYSIN **                           22510000 COUNT
                                         2251*010 PRINCE HR   SZ=20
                                         2251*020 EMP-NAME HR
                                         2251*030 SALARY F$ SZ=10 HF
                                        USE * DROP '62'
03 ** SYSIN **                           2261*030 SALARY SZ=10 F$
                                        END
02 ** SYSIN **                          USE * RENUMBER 8
03 ** SYSIN **                          END
                                         228001   SALARY / COUNT  AVERAGE-SAL
02 ** SYSIN **                          END
01 ** SYSIN **                          END
▲      ▲
|      |
|      └──────────── Indicates source of code
|
└─────────────────── Indicates level of nesting
```

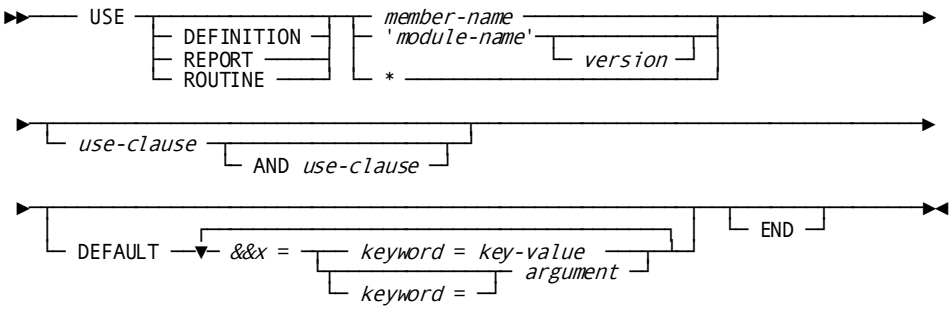
USE parameter syntax appears on the following page.

USE Parameter

Purpose

Permits the nesting of USE syntax within stored code.

Syntax



Syntax Rules

USE

Identifies a USE parameter. This keyword must precede all other clauses associated with this parameter. USE cannot appear in code that specifies =COPY or =MACRO parameters.

DEFINITION/REPORT/ROUTINE

These are optional identifiers that are for documentation purposes only. If one of these identifiers is used, it must follow USE and precede *member-name* or *module-name* (see below).

member-name

The name by which the stored code is accessed, as follows:

- **Under z/OS**, *member-name* is a member of a partitioned data set (PDS), CA Librarian library, or CA Panvalet library.
- **Under z/VSE**, *member-name* is either a book name of a source statement library (SSL) or a member of a CA Librarian library or CA Panvalet library.
- **Under z/VM**, *member-name* is a member of a z/VSE MACLIB or a z/OS partitioned data set.

The code must be stored in a library before run time and must be made available to the precompile phase of CA Culprit processing. The code must be stored with a record length of 80 bytes.

If the PARMLIB= option is coded on a PROFILE parameter that appears before the USE parameter, the parameters are copied from the library specified by the PARMLIB= option. Otherwise, the parameters are copied from the library selected as the default for PARMLIB at installation.

Note: For more information on system defaults, see the *CA IDMS Installation Guide* for your operating system.

module-name

(IDD only) Specifies a 1- to 32-character alphanumeric expression, enclosed in quotation marks, which identifies the module containing stored code.

version

Specifies the module version number; if no value is specified, CA Culprit uses the highest version number known to the data dictionary for *module-name*.

*****

This modifies the input stream parameters that immediately follow the USE parameter. An END clause signals the end of code modified by USE *.

DEFAULT

Identifies a DEFAULT clause. One DEFAULT clause is permitted with each USE parameter. In stored code, the DEFAULT clause should be the first line of code; in inline code, the DEFAULT clause should follow the USE parameter and associated WITH VALUES, CHANGE, DROP/KEEP, and RENUMBER clauses.

&&x =

Represents a symbolic parameter in stored or inline code. Symbolic parameter names are composed of && followed by a single alphanumeric character in the range 1 through 9 or A through Z. The symbolic parameter must be followed by an equal sign; no spaces surround the equal sign. Symbolic parameter names in the form &&x must be entered sequentially, starting with &&1.

This is followed by *keyword = key-value* or *keyword = argument*, a 1- to 72-character alphanumeric expression that associates a symbolic parameter with a keyword expression or argument.

The following rules apply to coding keyword and argument expressions on a DEFAULT clause:

- As many as 35 keyword expressions or arguments can be specified in a DEFAULT clause.
- The keyword or argument expression must appear entirely on one line, and must be separated from other expressions by either a comma or a space.
- Keyword values or arguments that contain embedded spaces, commas, slashes, or parentheses must be enclosed in quotation marks.

keyword =

A keyword associated with a symbolic parameter. If a WITH VALUES clause contains a keyword expression, the keyword definition is expected on the DEFAULT clause. *Keyword* is optional if the WITH VALUES clause contains arguments. (The WITH VALUES clause is discussed later in this chapter.)

key-value

Specifies a default value for the symbolic parameter. Default values are substituted for all symbolic parameters that are not assigned keyword values in the WITH VALUES clause.

argument

Supplies a default value for the symbolic parameter. Default values are substituted for all symbolic parameters that are not assigned argument values in the WITH VALUES clause. If no value for a symbolic parameter is defined in a DEFAULT clause, a null character string is used as the default.

use-clause

Specifies one of four optional clauses associated with the USE parameter that modify the copied or inline code. The four clauses are WITH VALUES, CHANGE, DROP/KEEP, and RENUMBER

Each clause is discussed separately later in this chapter.

AND

Signals the end of an action or clause and begins a new action or clause. AND links two clauses if it appears between two USE parameter clauses; that is, a clause that specifies WITH VALUES, CHANGE, DROP/KEEP, or RENUMBER. Any number of USE parameter clauses can be joined by the keyword AND, as shown in the example below:

```
USE * DROP 4 AND RENUMBER 7 AND CHANGE 'DEBIT' TO 'CREDIT'
```

In this example, AND links USE parameter clauses that drop type 4 edit parameters, renumber type 7 process parameters, and replace a character string.

When a clause action follows the keyword AND, AND signals the end of a clause action and the start of another action associated with the same clause, as shown in the example below:

```
USE * DROP 4 AND FIELD DEPARTMENT
```

In this example, AND links an action to drop all type 4 edit parameters and an action to drop the parameter that defines DEPARTMENT. Any number of actions can be linked by using the keyword AND.

END

Signals the end of a portion of inline CA Culprit code processed by a USE * parameter. No additional information, except comments, can appear with an END clause.

Usage

Coding Considerations

Special coding considerations that apply only to the USE parameter are as follows:

- **Quotation marks** can be single (') or double ("). Quotation marks are treated in the following manner:
  - Matching quotation marks of either variety are used to enclose an expression that contains spaces, commas, slashes, or parentheses; for example, "CURRENT BALANCE".
  - Quotation marks of one type are used to enclose an expression that must retain quotation marks of the other type; for example, "'ERROR MESSAGE'" is interpreted by CA Culprit as 'ERROR MESSAGE'.
  - Matching quotation marks of either type with no intervening spaces represent a null value; for example, '' or "".

- **Free-form coding** applies to the USE parameter. The parameter can be coded in any column in the range 1 through 72 and can extend over any number of lines; continuation lines must not specify an asterisk in column 1.
- **Comments** preceded by a dollar sign ($) can appear on any line within the USE parameter. Comments must appear entirely on one line.

More information:

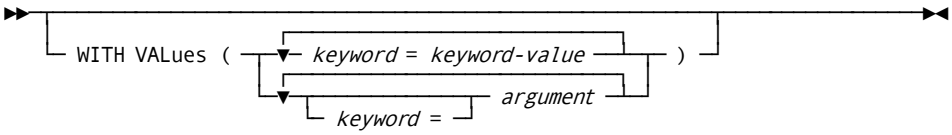
PROFILE Parameter (see page 29)
Storing and Accessing CA Culprit Code (see page 371)

WITH VALUES Clause

Purpose

The WITH VALUES clause assigns values to symbolic parameters in copied or inline code by means of keyword expressions or arguments.

Syntax



Syntax Rules

WITH VALUES

Identifies a WITH VALUES clause. One WITH VALUES clause is permitted with each USE parameter; the WITH VALUES clause should be coded before any CHANGE, DROP/KEEP, and RENUMBER clauses associated with the USE parameter.

keyword =

Specifies a keyword expression or argument enclosed in parentheses, or a list of keyword expressions or arguments enclosed in parentheses.

Keyword expressions and arguments assign values to symbolic parameters in copied or inline code. Keyword expressions can be submitted in any order in the WITH VALUES clause; arguments must be submitted in an order that corresponds sequentially to the symbolic parameters.

Keyword expressions and arguments are 1- to 72-character alphanumeric expressions.

keyword-value

Substitutes a value for a symbolic parameter in inline or stored code. *Keyword* is a keyword associated with the symbolic parameter through information coded on a DEFAULT clause.

If a value is not assigned to a symbolic parameter on the WITH VALUES clause, the value defined on the DEFAULT clause is used.

argument

Substitutes a value for a symbolic parameter in inline or stored code; the first argument in a list is assigned to symbolic parameter &&1, the second to &&2, and so on. If a default value specified on a DEFAULT clause is to be assigned to a symbolic parameter, code a comma in place of the argument in the argument stream.

Usage

The following rules apply to coding keyword expressions and arguments on a WITH VALUES clause:

- As many as 35 keyword expressions or arguments can be specified in a WITH VALUES clause.
- A WITH VALUES clause can specify either keyword expressions or arguments, but not both.
- A WITH VALUES clause can extend over any number of lines.
- The keyword expression or argument must appear entirely on one line and must be separated from other keyword expressions or arguments by a comma or a space.
- A keyword value or argument that contains embedded spaces, commas, slashes, or parentheses must be enclosed in quotation marks.

If the value substituted for the symbolic parameter is shorter or longer than the 3-character symbolic parameter, the length of the record is adjusted to accommodate the length of the substituted value. If the adjusted length is greater than 72 characters, continuation lines, with a maximum of 256 characters can be created to accommodate the data.

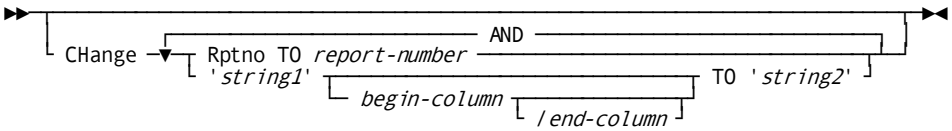
CHANGE Clause

Purpose

Modifies copied and inline code, as follows:

- The report number of report-specific CA Culprit parameters can be changed.
- Character strings can be replaced.

Syntax



Syntax Rules

CHANGE

Identifies a CHANGE clause. One or more CHANGE clauses can be coded for each USE parameter.

RPTNO TO *report-number*

Specifies a new report number for all report-specific parameters. *Report-number* must be a number in the range 0 through 99. The report number coded in columns 2 and 3 of all report-specific parameters changes to the number specified by *report-number*.

Only one CHANGE clause associated with a USE parameter can specify RPTNO TO.

string-1* TO *string-2

Replaces a character string within a specified column range.

If a range of columns is specified on a CHANGE clause that modifies a character string, the changes apply only to the first line of a CA Culprit parameter and not to associated continuation lines. Otherwise, the changes apply to all lines associated with the CA Culprit parameter.

string-1

Specifies an alphanumeric character string that is to be changed. *string-1* is a 1- to 70-character value, enclosed in quotation marks.

begin-column

Identifies the starting column of a range. *Begin-column* must be a number in the range 1 through 72; the default is 1.

If *begin-column* is specified, character string changes occur only within the specified range of columns.

end-column

Specifies the last column of a range. *End-column* must be a number in the range 1 through 72; the default is 72. The value specified for *end-column* must be equal to or greater than the value specified for *begin-column*. If specified, *begin-column* and *end-column* must be separated by a slash.

string-2

Specifies an alphanumeric expression that replaces the original string. *String-2* is a 1- to 70-character value, enclosed in quotation marks.

If the replacement string is a different length than the original string, the record length is adjusted to accommodate the length of the new string. If the adjusted record length exceeds 72 characters, one or more continuation lines, with a maximum of 256 characters can be created to accommodate the new string. The entire replacement string must appear on one line.

AND

Signals the end of one CHANGE action and the start of another as in the following example:

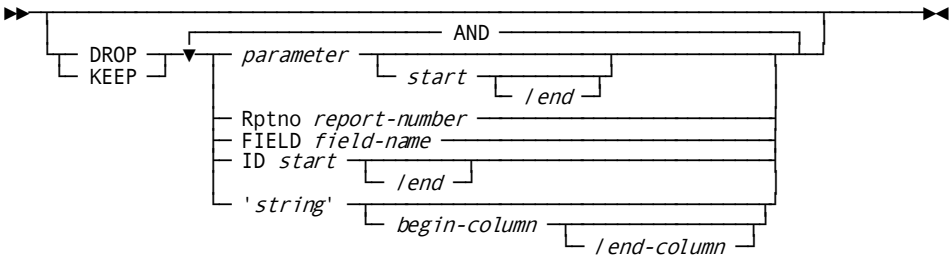
```
CHANGE 'CAT' TO 'DOG' AND 'MOUSE' 4/10 TO 'COW'
```

DROP/KEEP Clause

Purpose

Selectively drops or keeps CA Culprit parameters in inline or copied code depending upon parameter type, report number, field name, identification information, or character strings.

Syntax



Syntax Rules

DROP/KEEP

Identifies a DROP or KEEP clause. DROP eliminates the specified parameter from the parameter stream; KEEP retains the specified parameter and eliminates all other parameters. One or more DROP and KEEP clauses can be associated with a USE parameter.

When a parameter meets the criteria of both a DROP clause and a KEEP clause, precedence is given to DROP. Continuation lines associated with a parameter are also dropped or kept except for DROP/KEEP instructions that specify either the ID option or a column range on a string expression.

parameter start/end

Specifies that CA Culprit parameters are dropped or kept.

parameter

Specifies a valid CA Culprit parameter type:

Parameter	Synonym
INPUT	IN
REC	
SELECT	SEL
BYPASS	BYP
OUTPUT	OUT
SORT	
0	W
1	
3	T
4	H
5	D
6	S
7	I
8	B

start

Specifies the start of a range of edit or process parameters, as follows:

- If *parameter* specifies an edit parameter (4, 5, or 6), *start* must be an edit line number in the range 1 through 8.
- If *parameter* specifies a process parameter, (7 or 8), *start* must be a sequence number in the range 1 through 999.

If *start* is not specified, all parameters that specify *parameter* are dropped or kept.

end

Specifies the end of the range of edit line numbers or process sequence numbers. This value must follow a slash and be equal to or greater than *start*; the default is *start*.

RPTNO *report*

Specifies that all parameters with *report* coded in columns 2 and 3 are to be dropped or kept. *Report* must be a number in the range 0 through 99.

FIELD *field-name*

Specifies that any parameters that define *field-name* are to be dropped or kept; *field-name* specifies a 1- to 32-character alphanumeric value that is defined on a REC or work field parameter.

When more than one work field is defined for a single type 0 or type 1 work field parameter, the parameter is dropped or kept only if *field-name* is the first field defined on the parameter. In the following example, CA Culprit will ignore the request to keep or drop the record if DATA-FIELD is specified. If AGE-FIELD is specified, the requested action will occur.

```
010 AGE-FIELD NAME-FIELD DATA-FIELD
```

ID *start/end*

Specifies that parameters within a range of ID values are to be dropped or kept.

When ID is specified, DROP/KEEP criteria apply only to the first line of the parameter, and not to continuation lines.

start

A 1- to 8-character alphanumeric expression specifying a partial or complete identifier coded in columns 73 to 80.

end

A 1- to 8-character alphanumeric expression specifying a partial or complete identifier coded in columns 73 to 80. *End* must follow a slash and be equal to or greater than *start*; the default is *start*.

string begin-column/end-column

Specifies that parameters containing an identified character string within a column range are to be kept or dropped.

If a DROP/KEEP clause that specifies a character string also specifies a column range, DROP/KEEP criteria apply only to the first line of the parameter, not to any continuation lines. If the clause does not specify a column range, all lines associated with the parameter are dropped or kept.

string

A 1- to 70-character alphanumeric value, enclosed in quotation marks.

begin-column

Identifies the starting column of a range. This value must be a number in the range 1 through 72; the default is 1.

end-column

Specifies the last column of the specified range. This value must be a number in the range 1 through 72 preceded by a slash and equal to or greater than *begin-column*; the default is 72.

AND

Signals the end of one action on a DROP or KEEP clause and the start of another as shown in the example below:

```
DROP FIELD EMPLOYEE AND 'SALARY' 6/15.
```

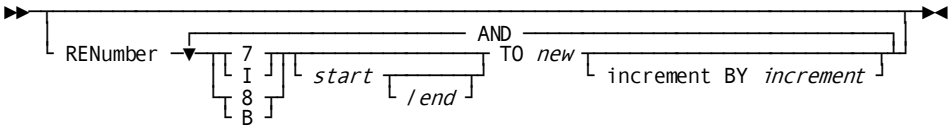
RENUMBER Clause

Purpose

Changes the sequence numbers on type 7 and type 8 process parameters.

Parameters that do not have sequence numbers are not assigned new sequence numbers; however, these parameters do maintain their internal relationship to parameters that specify sequence numbers. Sequence numbers specified on parameters that pass control to other process parameters are changed to the appropriate new values.

Syntax



Syntax Rules

RENUMBER

Specifies a RENUMBER clause; one or more RENUMBER clauses can be associated with a USE parameter. The RENUMBER clause should be coded after any WITH VALUES, CHANGE, and DROP/KEEP clauses associated with the USE parameter.

7/8

Specifies a type 7 or type 8 process parameter, respectively. A least one space must follow this specification.

start

Specifies the beginning of a range of sequence numbers to be renumbered. *Start* must be a value in the range 0 through 999; the default is 0.

end

Specifies the end of a range of sequence numbers to be renumbered. *End* must be a value in the range 0 through 999, preceded by a slash, and equal to or greater than *start*; the default is 999. During execution, only those parameters with sequence numbers within the specified range are renumbered.

new

Specifies the starting value of the new sequence numbers. *New* must be a number in the range 0 through 999; the default is 1. During execution, process parameters are renumbered within the specified range, starting with *new*.

BY *increment*

Specifies an increment value for renumbering. *Increment* must be a number in the range 1 through 999; the default is 1.

AND

Signals the end of a RENUMBER action and the start of another, as shown in this example:

```
RENUMBER 7 001/100 TO 5 BY 10 AND 8 001/020
```

Examples

Sample USE parameters and associated clauses are shown and described below.

Example 1: USE Parameter

```
USE DEFINITION CUSTFILE
USE REPORT CUSTLIST
```

The first USE parameter copies parameters stored in library member CUSTFILE. The second USE parameter copies parameters stored in library member CUSTLIST.

Example 2: WITH VALUES Clause

```
USE REPORT 'CUSTLIST' WITH VALUES
           (AMOUNT=50
            NAME=BROWN
            SALES=10000)
```

This USE parameter copies parameters stored in IDD. Keywords are used to assign values to symbolic parameters in the stored code. The keywords are associated with the symbolic parameters by using information coded on the DEFAULT clause. The values specified on the WITH VALUES clause override any specified default values.

Example 3: WITH VALUES Clause

```
USE CUSTFILE WITH VAL (50, BROWN, , 'THREE PIECE SUITS')
```

In this example, argument values replace symbolic parameters in the code stored in library member CUSTFILE. &&1 is assigned a value of 50; &&2 is assigned a value of BROWN; &&3 is assigned a value specified on a DEFAULT clause. The fourth argument is enclosed in quotation marks because it contains embedded spaces.

Example 4: CHANGE Clause

```
    -                     ◄-Inline code
    -
USE *
    CHANGE RPTNO TO 12 AND
           '52*010' 4/9  TO '410010' AND
           'CLIENT-NAME' TO 'NAME'
 113CLIENT ACCOUNTS
 1151*010 CLIENT-NAME
 1152*010 'SALES DISTRICT'
    -                     ◄-More inline code
    -
END
    -
    -                     ◄-More inline code
    -
```

In this example, the USE parameter modifies inline code that falls between the USE parameter and the END clause.

Report number 11 is changed to Report number 12. Columns 4 through 9 of a type 5 edit parameter are changed as follows: the detail line becomes a header line and the relative column number is changed to an absolute column number. Additionally, CLIENT-NAME changes to NAME within columns 1 through 72 on any records that specify this field name. The length of the records that specify CLIENT-NAME is adjusted to accommodate the shorter value.

Example 5: DROP Clause

```
USE * DROP FIELD SALARY
      DROP ID 20000100/20000175
 REC SALARY 1 10 2 DP=2
 REC NAME  11 20
 0151*010 NAME
 0151*020 SALARY                                    20000100
 0161*010 'TOTAL COMPENSATION FOR'                  20000125
 0162*010 'FACULTY PROFESSORS'                      20000150
 0162*020 SALARY SZ=11 DP=2                         20000175
END
```

In this example, inline code that appears between USE and END is modified by the DROP clauses. The first DROP clause drops the REC parameter that defines SALARY. The second DROP clause instructs CA Culprit to drop all records with ID values in the range 20000100 through 20000175; that is, the detail line for SALARY and all the type 6 edit parameters coded for Report 01. The modified code generates a report that lists the names of the faculty professors.

Example 6: DROP/KEEP Clause

```
USE *
KEEP 'SALARY' AND 'NAME'
DROP ID 20000100
 REC SALARY 1 10 2 DP=2
 REC NAME  11 20
 0151*010 NAME
 0151*020 SALARY                                        20000100
 0161*010 'TOTAL SALARY'
 0161*020 SALARY SZ=11 DP=2
END
```

The KEEP clause instructs CA Culprit to eliminate all parameters except for those that specify the character strings SALARY and NAME. The DROP clause instructs CA Culprit to eliminate the parameter with ID 20000100; this parameter also specifies the character string, SALARY.

When a parameter meets both DROP and KEEP criteria, the DROP clause is executed. The code that results from the instructions listed above is shown below. The code will cause a run-time error because SALARY must appear on a SORT parameter or on a type 5 edit parameter in order to appear on a type 6 edit parameter:

```
REC SALARY 1 10 2 DP=2
REC NAME  11 20
0151*010 NAME
0161*020 SALARY SZ=11 DP=2
```

Example 7: RENUMBER Clause

```
USE REPORT AVERAGE WITH VALUES (COUNT=12)
           RENUMBER 7 AND
                    8 50/100 TO 5 BY 10
```

Library member AVERAGE contains CA Culprit parameters that perform a procedure to compute averages. A symbolic parameter in the code is assigned a value of 12; the keyword COUNT is associated with the symbolic parameter through information coded on the DEFAULT clause in the stored code.

With the first RENUMBER clause, all sequenced type 7 parameters are renumbered starting with 001. The second RENUMBER action renumbers type 8 parameters with sequence numbers in the range 50 through 100; the new sequence numbers begin with 5 and increase by 10.

Example 8: DEFAULT

```
USE AGERPT                          ◄-Primary input
WITH VALUES (AMOUNT=50)

DEFAULT &&1=AMOUNT=100     ◄-Copy library member AGERPT
        &&2=PERIOD=365
        &&3=START=010184
        &&4=END=123184
 010 AMT &&1  PERIOD &&2
*    START-DATE &&3  FINISH-DATE &&4
         -
         -                        ◄-More stored code
         -
```

Library member AGERPT contains a report on aging; default values are defined for four symbolic parameters in the stored code. The USE parameter accesses the code stored in AGERPT; the default values defined on the DEFAULT clause replace all symbolic parameters within the stored code, with the exception of &&1, which is assigned a value of 50 in the keyword expression coded on the WITH VALUES clause.

=COPY Parameter Overview

What It Does

The =COPY parameter is used to copy stored code and to modify copied or inline code.

The following types of modifications are available:

- A report number in copied or inline code can be changed.
- An INPUT parameter in stored code can be suppressed.

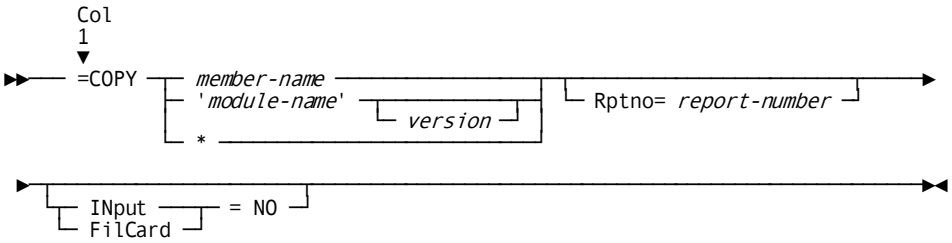
Modifications made to inline code apply to all subsequent parameters until another =COPY or =MACRO parameter is encountered. Modifications made to copied code apply only to parameters residing in a member of the stored library or in a book member. In all cases, the modifications apply only to the CA Culprit run for which they were specified.

=COPY Parameter

Purpose

Copies stored code and modifies copied or inline code.

Syntax



Syntax Rules

=COPY

Identifies an =COPY parameter. It must be coded starting in column 1.

The entire statement must be contained on one line. One or more spaces must separate =COPY from the variable information that follows. =COPY cannot be specified in a report that specifies the USE parameter. The =COPY parameter cannot be nested; therefore, it cannot be specified in stored code.

member-name/module-name/*

Identifies the source of code to be modified.

member-name

The name by which the stored code is accessed, as follows:

- **Under z/OS**, the name identifies a member of a partitioned data set (PDS), CA Librarian library, or CA Panvalet library.
- **Under z/VSE**, the name identifies a book of a source statement library (SSL), a member of a CA Librarian library, or a member of a CA Panvalet library.
- **Under z/VM**, *member-name* is a member of a z/VM MACLIB or a z/OS partitioned data set.

The accessed code must be stored in a library prior to run time and made available to the precompile phase of CA Culprit processing. The code must be stored with a record length of 80 bytes.

If the PARMLIB= option is coded on a PROFILE parameter that appears before the USE parameter, the parameters are copied from the library specified on the PARMLIB= option. Otherwise, the parameters are copied from the library selected as the default for PARMLIB at installation time.

Note: For more information on system defaults, see the *CA IDMS Installation Guide* for your operating system.

module-name

(IDD users only) Specifies a 1- to 32-character alphanumeric expression, enclosed in single quotation marks, that identifies the module containing stored code.

version

Specifies the module version number; if no value is specified, CA Culprit uses the highest version number known to the data dictionary for *module-name*.

*****

Indicates that parameters entered directly in the input parameter stream are to be modified. The =COPY parameter applies to all parameters immediately following it up to the next =COPY or =MACRO parameter (if any).

RPTNO = *report-number*

A keyword expression that changes the report number coded in columns 2 and 3 of all inline or copied parameters to the number specified by *report*. *Report-number* must be a number in the range 00 through 99.

Only one report number can be associated with an =COPY parameter. Parameters that do not contain a report number, such as input definition parameters, are not affected by an =COPY parameter that specifies RPTNO=.

INPUT = NO

A keyword expression that suppresses delivery of the INPUT parameter in copied code. If this expression is used, an alternative INPUT parameter must be supplied before the =COPY parameter.

FILCARD may be used as a synonym for INPUT.

Examples

Sample =COPY parameters are shown and described below.

The following report shows a sample Sequential Parameter Listing created by CA Culprit during the precompile phase in response to an =COPY parameter. The listing prints plus signs (+) before each line of copied code.

```
mm/dd/yy                    SEQUENTIAL PARAMETER LISTING   Vnn.n  PAGE     1
00 ** SYSIN **                        IN 80
  INSTALLATION SECURITY OPTION IS NO
                                      =COPY RECS                              ┐
                                    +  REC EMP-NAME      1   25   'EMPLOYEE' 'NAME'  │   Parameters copied
                                    +  REC EMP-LNAME    11   10                      │   from library member
                                    +  REC DEPARTMENT   30   20                      ├── RECS
                                    +  REC TITLE        50   20                      │
                                    +  REC SALARY       70   10  3  DP=2            │
                                       01SORT DEPARTMENT,1,TITLE                     ┘
                                       01410010 DEPARTMENT
                                       0151*010 TITLE
                                       0151*020 EMP-NAME
                                       0151*030 SALARY
```

The following report shows the Input Parameter Listing generated after the sample Sequential Parameter Listing. On this listing, no distinction is made between parameters included by the =COPY parameter and those in the input parameter stream.

```
mm/dd/yy                    INPUT PARAMETER LISTING        Vnn.n PAGE     1
********
INPUT    RECORD TYPE BLOCK  FILE DESCRIPTION...
********
 INPUT   00080   F
********
REC      START SIZE TYPE DP FIELD-NAME                    RECORD-NAME,LEVEL
********
 REC     00030  020         DEPARTMENT
 REC     00011  010         EMP-LNAME
 REC     00001  025         EMP-NAME                        'EMPLOYEE'     'NAME'
 REC     00070  010   3   2 SALARY
 REC     00050  020         TITLE
********
SORT     BREAK A/D  SORT FIELD-NAME
********
 01 SORT   1        DEPARTMENT
                    TITLE
********
EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
********
 01  4     1      0010   DEPARTMENT
********
EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
********
 01  5     1      *010   TITLE
 01  5     1      *020   EMP-NAME
 01  5     1      *030   SALARY
 EXTRACT WILL BE PERFORMED
 PROFILE OPTION IN EFFECT: RELEASE =  6
```

Example 1

```
=COPY PARAMETERS RPTNO=14
```

CA Culprit parameters are copied from library member PARAMETERS. The report number of all report-specific parameters is changed to 14.

Example 2

```
=COPY * RPTNO=02
 060UT D
 063'STUDENT ACCOUNTS'
 0651*010 NAME
      -                    ◄More inline code
      -
=MACRO *
=MEND
      -
      -                    ◄-More inline code
```

In this example, the inline code between the =COPY and =MACRO parameters is modified. Report number 06 is changed to 02.

Example 3

```
 INPUT 80 F 400 PS(3375)
=COPY INPUTPARAMS IN=NO
```

Except for any INPUT parameters, all of the parameters contained in library member INPUTPARAMS are copied. An alternative INPUT parameter is provided before the =COPY parameter.

More information:

PROFILE Parameter (see page 29)
Storing and Accessing CA Culprit Code (see page 371)

=MACRO Parameter Overview

=MACRO parameters copy stored code and modify inline and copied code.

The following types of modifications are available using =MACRO and its associated parameters, =DROP and =CHANGE:

- Arguments can replace symbolic parameters.
- Parameters can be suppressed.
- Report numbers can be changed.
- Sequence numbers of type 7 or 8 process parameters can be changed.
- Line numbers of edit parameters can be altered.
- Output field positions on edit parameters can be changed.

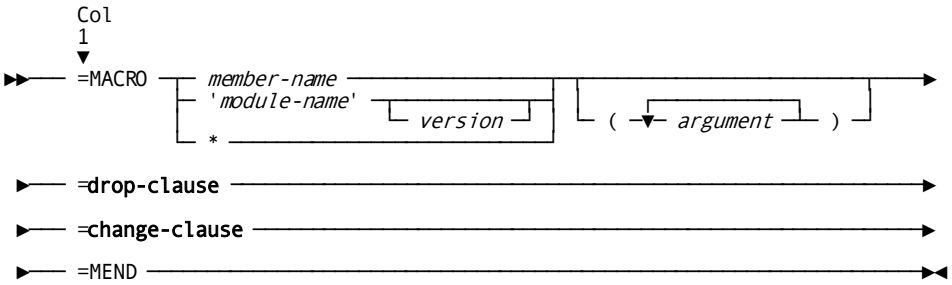
Modifications made to inline code apply to all subsequent parameters until another =COPY or =MACRO parameter is encountered. Modifications made to stored code apply only to parameters contained in the stored member or book. In all cases, modifications apply only to the CA Culprit run for which they were specified.

=MACRO Parameter

Purpose

Copies stored code and modifies inline and copied code.

Syntax



Syntax Rules

=MACRO

Identifies the =MACRO parameter. It is coded starting in column 1 and is followed by one or more spaces.

member-name/module-name/*

Identifies the source of code to be modified. library

member-name

The member name by which the stored code is accessed, as follows:

- **Under z/OS or** , the name is a member of a partitioned data set (PDS), CA Librarian library, or CA Panvalet library.
- **Under z/VSE**, the name is a book of a source statement library (SSL), a member of a CA CA

Parameters must be stored in a library prior to run time and made available to the precompile phase of CA Culprit processing. The code must be stored with a record length of 80 bytes.

If the PARMLIB= option is coded on a PROFILE parameter that appears before the USE parameter, the parameters are copied from the library specified on the PARMLIB= option. Otherwise, the parameters are copied from the library selected as the default for PARMLIB at installation.

Note: For more information on system defaults, see the *CA IDMS Installation Guide* for your operating system.

module-name

(IDD users only) Specifies a 1- to 32-character alphanumeric expression, enclosed in single quotation marks that identifies the module containing stored code.

version

Specifies the version number of the module. If no value is specified, CA Culprit uses the highest version number known to the data dictionary for *module-name*.

*****

Indicates that parameters placed directly in the input parameter stream are to be modified. The =MACRO parameter applies to all parameters that immediately follow the =MEND parameter up to the next =COPY or =MACRO parameter, if any.

argument

Specifies a single value or a list of values enclosed in parentheses; each value is a 1- to 53-character alphanumeric value that replaces a symbolic parameter in copied code. Up to 35 arguments can be specified.

The order of the listed arguments is important; the first argument is assigned to symbolic parameter &&1, the second to &&2, and so on. The sequence of symbolic parameters is &&1 through &&9 followed by &&A through &&Z. Symbolic parameters that appear in comments or in fixed-format portions of a parameter are not replaced. An =MACRO parameter that contains symbolic parameters is illustrated under examples.

If the value of the symbolic parameter is shorter or longer than the 3-character symbolic parameter, the length of the record is adjusted to accommodate the length of the substituted value. If the length of the record exceeds 72 characters, one or more continuation lines, to a maximum of 256 characters, are created to accommodate the data.

The following rules apply to coding a list of arguments on an =MACRO parameter:

- As many as 35 arguments can be specified over any number of lines.
- Each argument must appear on one line and must be separated from other arguments by a comma or a space.
- Arguments that contain embedded spaces, commas, slashes, apostrophes, or parentheses must be enclosed in quotation marks. To print an apostrophe, code two single quotation marks; for example, 'ERNIE''S'.

=drop-clause

Specifies an optional parameter associated with the =MACRO parameter. This parameter is discussed separately later in this chapter.

=change-clause

Specifies an optional parameter associated with the =MACRO parameter. This parameter is discussed separately later in this chapter.

=MEND

Signals the end of an =MACRO parameter. Each =MACRO parameter must have a corresponding =MEND parameter. Any number of =DROP and =CHANGE clauses can be entered between an =MACRO and an =MEND parameter.

=MEND must be coded starting in column 1; no other information may appear on the same line. Parameters that are modified inline by an =MACRO * instruction must appear immediately after the =MEND parameter.

Usage

The following coding considerations apply to the =MACRO parameter:

- An =MACRO parameter can continue from one line to the next. An asterisk (*) is required in column 1 of each continuation line.
- An =MACRO parameter is limited by any one of the following conditions:
  - A maximum of 40 lines of code
  - A maximum of 400 characters
  - A maximum of 100 keywords, delimiters, and operands

    Note: Within each =MACRO parameter (that is, =MACRO and any associated parameters), a maximum of 20 keyword expressions (that is, REC=, TYPE=, CNST=, OLD=, and NEW=) is permitted. Each occurrence of a keyword expression must appear entirely on one line.
- Each =MACRO parameter must end with an =MEND parameter; optional parameters (that is, =DROP or =CHANGE) must be specified between an =MACRO and an =MEND parameter.
- Comments must be coded immediately after the =MACRO statement and before any =MACRO clauses:

```
=MACRO *
* $THIS IS A COMMENT
=DROP REC=TITLE
=MEND
```

Example

The following is a sample =MACRO statement with symbolic parameters.

Five symbolic parameters (&&1 through &&5) appear in the CA Culprit parameters associated with the =MACRO parameter. The arguments corresponding to the symbolic parameters are listed in parentheses on the =MACRO parameter:

```
  IN  80
  REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
  REC EMP-LNAME  11 15
  REC SALARY     70 10 3 DP=2
  REC DEPARTMENT 30 20         'DEPARTMENT'
 =MACRO * (DEPARTMENT 'BRAIN STORMING      ' EMP-NAME SALARY 60000)
 =MEND
  01SORT DEPARTMENT,-,EMP-LNAME
  013EMPLOYEE SALARIES
  010COUNT 1
  017010 &&1 NE &&2 DROP
  017    &&4 GE &&5 DROP
  017030 COUNT + 1 COUNT
  0151*010 &&3           HR
  0151*020 &&4     SZ=7  F2  HF
  0161*010 'TOTAL COUNT'
  0161*020  COUNT  SZ=8
```

More information:

PROFILE Parameter (see page 29)
Storing and Accessing CA Culprit Code (see page 371)

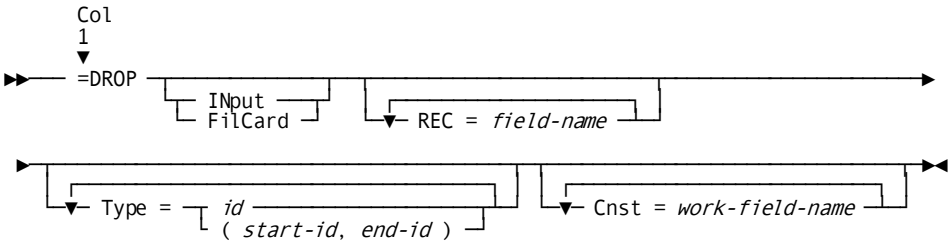
=DROP Clause

Purpose

Eliminates one or more of the following CA Culprit parameters from inline and copied code:

INPUT parameters	Title parameters
REC parameters	Edit parameters
SORT parameters	Process parameters
OUTPUT parameters	Work field parameters

Syntax



Syntax Rules

=DROP

Identifies an =DROP clause. Within each =MACRO parameter, =DROP clauses are processed before any =CHANGE clauses.

INPUT

Eliminates an INPUT parameter in copied or inline code. The user must supply an alternative INPUT parameter before the =MACRO parameter.

REC = *field-name*

A keyword expression that eliminates the REC parameter that defines the specified field in copied or inline code. *Field-name* must be specified as it appears on the REC parameter.

TYPE =

A keyword expression that suppresses delivery of all parameters that contain the specified parameter ID or that fall within the range of specified parameter ids, as follows:

id

Specifies a valid CA Culprit parameter type, as follows:

- A SORT parameter
- An OUTPUT (OUT) parameter
- A title (3 or T) parameter
- An edit parameter in the form *abcccc*, where *a* identifies the type of edit parameter, that is, 4 (H), 5 (D), or 6 (S); *b* identifies an optional edit line number in the range 1 through 8; and *cccc* identifies an optional column position
- A process parameter in the form *abbb*, where *a* identifies the type of process parameter, that is, 7 (I) or 8 (B); and *bbb* identifies an optional sequence number

Each inline or copied parameter, starting with column 4, is compared to the value of *id* for the length of *id*. When a match occurs, the inline or copied parameter is eliminated.

Note: If a process parameter with a sequence number is dropped, any process parameters that immediately follow and that do not specify sequence numbers are also dropped.

start-id, end-id

Specifies a range of edit or process parameters, enclosed in parentheses. The length of both values must be the same. A comma or at least one space must separate the two range values.

Each inline or copied parameter, starting with column 4, is compared with the range of values for the length of a range value. If the inline or copied parameter falls within the range of values inclusive, the parameter is eliminated.

CNST = *work-field-name*

A keyword expression that suppresses delivery of the specified work field defined on a work field parameter or on an implied subscript parameter. Other field names defined on the same work field parameter are retained.

Usage

The following coding considerations apply to the =DROP clause:

- =DROP is coded starting in column 1; =DROP instructions can continue from one line to the next. An asterisk (*) is required in column 1 of each continuation line.
- An =DROP clause is limited by one of the following conditions:
  - A maximum of 400 characters
  - A maximum of 100 keywords, delimiters, and operands.

    Note: Within each =MACRO parameter (that is, =MACRO and any associated parameters), a maximum of 20 keyword expressions (that is, REC=, TYPE=, CNST=, OLD=, and NEW=) is permitted. Each occurrence of a keyword expression must appear entirely on one line.
- An =DROP clause can appear one or more times within an =MACRO parameter.

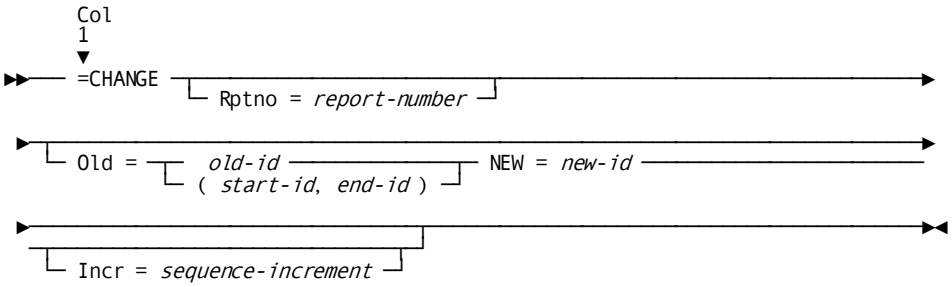
=CHANGE Clause

Purpose

Modifies copied or inline code:

- The report number of report specific parameters can be changed.
- The parameter type, line number, and field column position on edit parameters can be changed.
- The parameter type and sequence number on process parameters can be changed.

Syntax



Syntax Rules

=CHANGE

Identifies an =CHANGE clause. Within each =MACRO parameter, =CHANGE clauses are processed after any =DROP clauses.

Rptno = *report-number*

Changes the report number in columns 2 and 3 of report-specific parameters to the value specified by *report-number*. *Report-number* must be a number in the range 00 through 99.

No more than one RPTNO= keyword expression can be associated with a single =MACRO parameter. Parameters such as input definition parameters that do not specify report numbers are not affected by this instruction.

Old =

Specifies that edit or process parameters identified in the OLD= keyword expression are to be renumbered according to the value specified in the NEW= keyword expression.

old-id

Identifies an edit or process parameter to be changed, as follows:

- An edit parameter is specified in the form *abcccc*, where *a* identifies the type of edit parameter, that is, 4 (H), 5 (D), or 6 (S); *b* identifies an optional edit line number in the range 1 through 8; and *cccc* identifies an optional column position.
- A process parameter is specified in the form *abbb*, where *a* identifies the type of process parameter, that is 7 (I) or 8 (B); and *bbb* identifies an optional sequence number in the range 000 through 999.

Each copied or inline parameter, starting with column 4, is compared with the value of *old-id* for as many characters as are specified. When a match is found, the copied parameter is renumbered as specified by the NEW= keyword expression. If more than one OLD= keyword expression applies to a record, only the task associated with the first OLD= keyword expression is performed.

start-id, end-id

Identifies a range of edit parameters or process parameters to be renumbered. The range must be enclosed in parentheses. A comma or at least one space must separate the values specifying the range.

During processing, the copied or inline parameter is compared to the values specifying the range, beginning with column 4. When a match is found, the parameters within the range are renumbered as specified by the NEW= keyword expression (see below). If more than one OLD= keyword expression applies to the range of parameters, only the task associated with the first OLD= keyword expression is performed.

new-id

Modifies the value or values identified in the OLD= keyword expression. The value of *new-id* must refer to the same type of parameter (that is, an edit or a process parameter) referenced in the OLD= keyword expression. If *new-id* refers to a process parameter, it must specify a sequence number in the range 000 through 999.

The value of *new-id* replaces all parameters specified by the OLD= keyword expression, starting with column 4 of each parameter. For example, if NEW=51, then the value 5, which identifies a type 5 edit parameter, replaces the value in column 4, and the value 1, which identifies edit line number 1, replaces the value in column 5 for all edit parameters identified in the OLD= keyword expression.

Edit parameters that are to be renumbered can be submitted in any order since CA Culprit sorts the parameters before renumbering occurs. Process parameters with sequence numbers can also be submitted in any order; the following considerations apply if sequence numbers are modified:

- Parameters without sequence numbers retain their relationship to parameters with sequence numbers, but do not receive a sequence number. The first type 7 and type 8 process parameters in copied code should specify sequence numbers; these sequence numbers can act as delimiters for =CHANGE instructions.
- If a parameter specifies a sequence number as a result action, the sequence number is modified if the parameter that receives processing control is within the copied code. If the parameter that receives processing control is not in the copied code, the branch instruction is not modified, and the following warning message is printed:

  `RESULT FIELD nnn LEFT UNCHANGED`

Incr = *sequence-increment*

A keyword expression that specifies the increment value for renumbered process parameters. *Sequence-increment* must be a 3-digit number in the range 000 through 999; the default is 001.

Usage

The following coding considerations apply to =CHANGE:

- =CHANGE is coded starting in column 1; =CHANGE instructions can continue from one line to the next. An asterisk (*) is required in column 1 of each continuation line.
- An =CHANGE clause is limited by one of the following conditions:
  - A maximum of 400 characters
  - A maximum of 100 keywords, delimiters, and operands

    Note: Within each =MACRO parameter (that is, =MACRO and any associated parameters), a maximum of 20 keyword expressions (that is, REC=, TYPE=, CNST=, OLD=, and NEW=) is permitted. Each occurrence of a keyword expression must appear entirely on one line.
- An =CHANGE clause can appear one or more times within an =MACRO parameter.

Examples

Sample =MACRO parameters and associated parameters are shown and described below.

Example 1: =DROP

```
 INPUT 80
=MACRO 'PARMSET'
=DROP INPUT REC=FIELDA REC=FIELDC CNST=WORKFLD1
=MEND
```

PARMSET is the name of the module defined to IDD; it contains the set of parameters inserted into the CA Culprit parameter stream. =DROP deletes the INPUT record, REC parameters for FIELDA and FIELDC, and a work field parameter, WORKFLD1. =MEND completes the =MACRO parameter. An alternative INPUT parameter replaces the one dropped by the =MACRO parameter.

Example 2: =CHANGE

```
=MACRO PARMSET
=CHANGE OLD=51, NEW=52, OLD=7010, NEW=7500
=CHANGE OLD=(7100,7200)  NEW=(7600)  INCR=002
=MEND
```

In this example, the parameters stored in the library member PARMSET are modified for the CA Culprit run. The first =CHANGE converts all type 5 edit parameters that specify edit line 1 to parameters that specify edit line 2; a type 7 process parameter with sequence number 010 is renumbered with sequence number 500. The second =CHANGE clause renumbers type 7 process parameters with sequence numbers between 100 and 200. The new sequence numbers start with 600 and increase by 002.

An alternative way of coding the above example using one =CHANGE clause and a continuation line follows:

```
=CHANGE OLD=51  NEW=52, OLD=7010  NEW=7500
*OLD=(7100,7200)  NEW=(7600)  INCR=002
```

Example 3: =DROP/=CHANGE

```
 INPUT CARD
=MACRO PARMSET  (SORT-1,SORT-2)
=DROP  INPUT  TYPE=4  TYPE=(8000,8499)
=DROP TYPE=3
=CHANGE RPTNO=07
=MEND
```

SORT-1 and SORT-2 are arguments that replace symbolic parameters &&1 and &&2 stored in the parameters in library member, PARMSET. The =DROP clauses eliminate the following records from the code copied from PARMSET: the INPUT parameter, all type 4 edit parameters, the title parameter, and type 8 process parameters with sequence numbers in the range 000 to 499. Other type 8 parameters are retained, except for any unsequenced type 8 parameters that immediately follow sequence number 499. All remaining report-specific parameters are changed so that 07 appears in columns 2 and 3.

Example 4: =CHANGE Clause

```
=MACRO *
=CHANGE  RPTNO=02
=MEND
 0151*001  FIELD-1
 0151*002  FIELD-2
 -                ◄Inline parameters to be
 -                     modified
 -
 0151*026  FIELD-26
=MACRO *
=MEND
 -
 -              ◄-Other CA-Culprit parameters
 -
```

=MACRO * indicates that inline parameters are to be modified. The type 5 edit parameters for fields FIELD-1 through FIELD-26 are changed so that 0151 becomes 0251, reflecting a new report number. The second =MACRO * serves only to stop the effect of changes resulting from the preceding =MACRO parameter. The second =MACRO * parameter would not be necessary if the parameters to be changed occurred at the end of the parameter stream.

Example 5: =DROP Clause

```
=MACRO *
=DROP  TYPE=7010
=MEND
 227010  MOVE FIELD-A  TO  FIELD-B
 227     MOVE FIELD-B  TO  FIELD-C
 227     TAKE 1
 227020  TAKE 2
```

=MACRO * indicates that parameters to be modified follow inline. The type 7 process parameter containing sequence number 010 is dropped; the two type 7 parameters that immediately follow are also dropped because they do not specify sequence numbers. The only parameter that remains is the type 7 process parameter with sequence number 020.

=MACRO AMLIST Overview

What It Does

The =MACRO AMLIST routine is used to list the contents of data files. The listings provided by the AMLIST*n* routine can be used to review the contents or check the format and sequence of records on file before CA Culprit reports are run against the file.

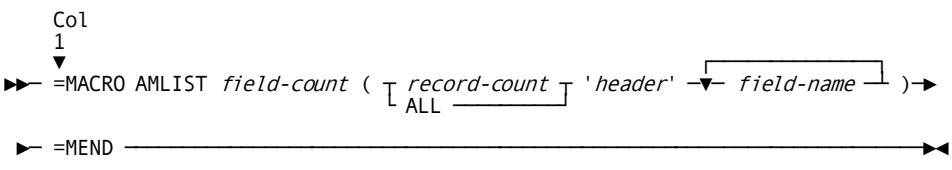
AMLIST*n* is stored in the CA Culprit source library at installation. The routine lists from three to ten fields from any file for the specified number of records. The routine prints records in the order they appear on the file, without totals. AMLIST*n* can be modified, however, to sort the records, select specified records, and print total lines.

=MACRO AMLIST

Purpose

Lists the contents of data files.

Syntax



Syntax Rules

=MACRO AMLIST

Identifies the list routine instruction. It must be coded starting in column 1.

field-count

Specifies the number of fields to be listed in the report. *Field-count* must be a number in the range 3 through 10.

record-count

Specifies the number of records to be printed; *record-count* must be a positive integer value.

ALL

Specifies that all records of the file are to be printed in the report.

'header'

A 1- to 53-character alphanumeric literal that specifies a header for the report. *'Header'* must be enclosed in single quotation marks.

field-name

Specifies the names of fields to be listed in the report. The field names must be defined on REC parameters. These names also serve as report column headers. The number of field names specified must be the same as the number coded for *field-count*.

=MEND

Signals the end of an =MACRO parameter. Each =MACRO parameter must have a corresponding =MEND parameter.

=MEND must be coded starting in column 1; no other information may appear on the same line.

Usage

The report generated by the =MACRO AMLIST parameter can be modified by coding CA Culprit parameters immediately following the =MEND instruction, as described below:

- A SORT parameter can be used to sort the records within the report.
- A type 7 process parameter can be used to select records to be output.
- Type 6 edit parameters can be used to generate totals for the report.

Usage

=MACRO AMLIST parameters can continue from one line to the next; an asterisk (*) is required in column 1 of each continuation line.

=CHANGE and =DROP clauses can follow an =MACRO AMLIST instruction; an =MEND instruction must follow.

Examples

Sample =MACRO AMLIST parameters are shown and described below.

Example 1

```
 REC EMP-NAME  1  25
 REC TITLE    50  20
 REC SALARY   70  10   3 DP=2
=MACRO AMLIST3 (10 'COMPANY EMPLOYEES' EMP-NAME TITLE
*               SALARY)
=MEND
```

The =MACRO AMLIST parameter outputs three input file fields: EMP-NAME, TITLE, and SALARY. The first ten records in the input file will print; the field names act as column headers. The report will contain detail information only; the header for the report is COMPANY EMPLOYEES.

The following two figures illustrate the Sequential Parameter Listing generated by the =MACRO AMLIST instruction and the report generated. The Sequential Parameter Listing displays a plus sign (+) to identify parameters copied by the AMLIST instruction. CA Culprit substitutes the argument values for the symbolic parameters; two plus signs (++) identify parameters for which a substitution has occurred.

Example 2

```
=MACRO AMLIST3 (10 'COMPANY EMPLOYEES' EMP-NAME TITLE
*               SALARY)
=CHANGE R=01
=MEND
 01SORT TITLE,-
 017010 SALARY GT 25000 DROP
 0161*030 SALARY
```

The =MACRO AMLIST parameter is the same as the one described in the above example except that the report generated by this statement is modified by the =CHANGE clause, the SORT parameter, the type 7 process parameter, and the type 6 edit parameter that follow the =MEND clause. In this example, the records print in alphabetical order by job title. Records with a value for SALARY greater than 25000 are not extracted. The total line prints the accumulated value for SALARY for each control break and at the end of the output phase.

Sequential Parameter Listing Using =MACRO AMLIST

```
mm/dd/yy                    SEQUENTIAL PARAMETER LISTING   Vnn.n  PAGE

00 ** SYSIN **                           IN 80
  INSTALLATION SECURITY OPTION IS NO
                                         REC EMP-NAME  1  25
                                         REC TITLE    50  20
                                         REC SALARY   70  10   3 DP=2
                                        =MACRO AMLIST3 (10 'COMPANY EMPLOYEES' EMP-NAME TITLE
                                        *               SALARY)
                                        =MEND
                                  ┌──+  03$00****Culprit ROUTINE-AMLIST3
                                  │  +  030 SEQUENCE ALL
                                  │  +  033DETAIL LIST
                                  │  +  034100010&&2
                                  │  ++ 034100010'COMPANY EMPLOYEES'
                                  │  +  03420001 ' '
            Parameters            │  +  0351*010 &&3           HF
            copied from           │  ++ 0351*010 EMP-NAME      HF
            library member ──────►│  +  0351*020 &&4           HF
            AMLIST3               │  ++ 0351*020 TITLE         HF
                                  │  +  0351*030 &&5           HF
                                  │  ++ 0351*030 SALARY        HF
                                  │  +  0368*001 ' '
                                  │  +  037110SEQUENCE A  1       SEQUENCE      $  COUNT RECORDS READ
                                  │  +  037           M  &&1      ALL           $  GET MAXIMUM
                                  │  ++ 037           M  10       ALL           $  GET MAXIMUM
                                  │  +  037   ALL     EQ 0        TAKE          $  TAKE ALL RECORDS
                                  │  +  037   SEQUENCE GT &&1     DROP          $  DROP IF OVER MAXIMUM
                                  └─ ++ 037   SEQUENCE GT 10      DROP          $  DROP IF OVER MAXIMUM
```

Report Generated Using =MACRO AMLIST

```
REPORT NO. 03              DETAIL LIST               mm/dd/yy PAGE     1
COMPANY EMPLOYEES
▲                              ┌─
|    Field names──────────────►| EMP-NAME                      TITLE                          SALARY
|    used as                   └─
|    column            TERRY     CLOTH                  HUMIDITY CONTROL CLK                    38,000.00
|    headers           PHINEAS   FINN                   KEEPER OF BALLOONS                      45,000.00
|                      JOE       KASPAR                 WINTERIZER                              31,000.00
|                      MARK      TIME                   WINTERIZER                              33,000.00
|                      ROGER     WILCO                  MGR THERMOREGULATION                    80,000.00
|                      ROY       ANDALE                 SNOWBLOWER                              33,500.00
|                      HARRY     ARM                    STURM/DRANG ADMIN                       46,000.00
|                      C.        BREEZE                 KEEPER OF THE WINDS                     38,000.00
|                      CAROLYN   CROW                   RAINDANCE CONSULTANT                    37,500.00
|                      BURT      LANCHESTER             RAINMAKER                               54,500.00
|
| RECORDS WRITTEN FOR REPORT 03─       16
|
|
|
└──────── Report header
```

Chapter 9: CA Culprit in the CA IDMS/DB Environment

This section contains the following topics:

Overview

Most of the information presented in this guide also applies to CA Culprit runs that access a CA IDMS/DB database. However, database access runs also include CA Culprit parameters that are unique to the CA IDMS/DB environment. These parameters are required to access the type of structures that are stored in a CA IDMS/DB database.

Non-SQL Defined Database

The following structures are associated with non-SQL defined databases in CA IDMS/DB:

- Database records
- Logical records
- Tables

Each type of record structure is discussed separately below, followed by a discussion of CA Culprit parameters unique for database access.

SQL Defined Database

CA Culprit can also access data in CA IDMS/DB in SQL defined databases:

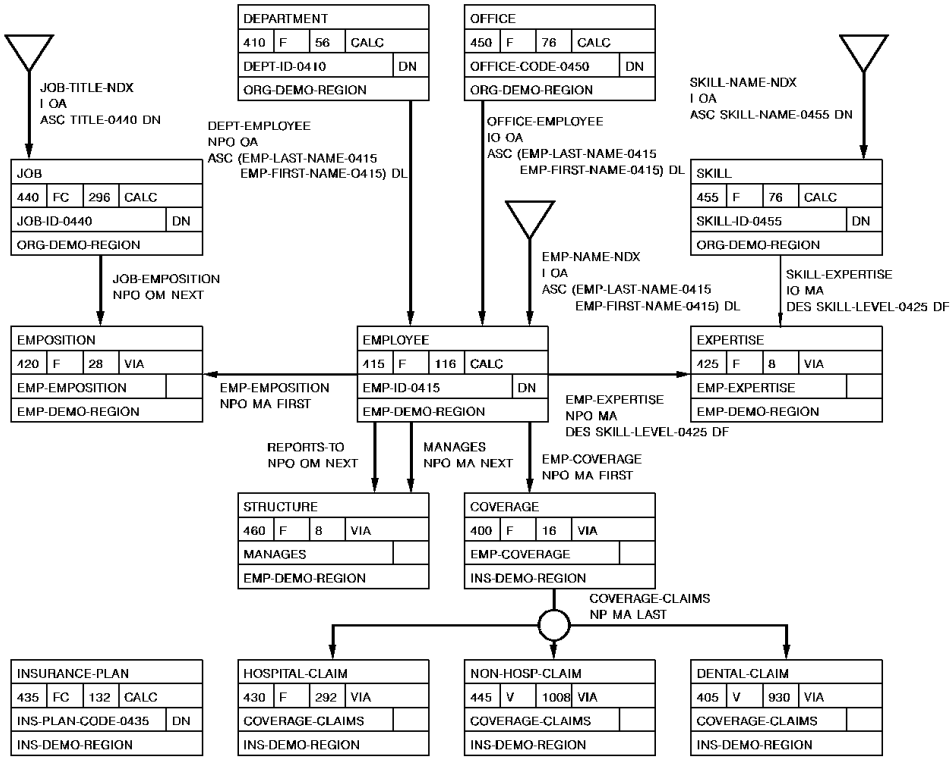
- Tables

More information:

Accessing SQL Defined Tables (see page 289)

Database Record Access

By using **database records**, the user can specify a path to navigate the a non-SQL defined database. The path consists of a string of one or more database records connected according to established set relationships. The following figure illustrates a non-SQL defined database structure in which the records participate in set relationships.



Note: CA Culprit navigates the database according to the set relationships that connect one record type to another.

CA Culprit can access records in a database by using an area sweep, that is, sequentially within an area or directly by using a CALC-key value, a db-key value, or an index-key value:

- A **CALC-key** value is a value of a designated field in the record, such as the employee ID field. In the figure given earlier, the EMPLOYEE, DEPARTMENT, OFFICE, JOB, SKILL, and INSURANCE-PLAN records can be accessed by using a CALC-key value.
- A **db-key** (database key) value identifies the physical address of the record in the database. Every record can be retrieved by using its db-key value.
- An **index-key** value is a value of a designated field in the record that has an index established on it; for example, the EMPLOYEE record is indexed according to employee last names. The SKILL and JOB records in the figure given earlier are also indexed.

Logical Record Access

By using logical records, the user can navigate a non-SQL defined database without knowing the structure of the database. A logical record is a concatenation of one or more database records that appears to the CA Culprit user as a single record. For example, the following report illustrates the subschema definition of the EMP-JOB-LR logical record, which includes the DEPARTMENT, EMPLOYEE, OFFICE, and JOB records shown in the figure earlier shown.

```
ADD
LOGICAL RECORD NAME IS EMP-JOB-LR
   ELEMENTS ARE
     EMPLOYEE
     DEPARTMENT
     JOB
     OFFICE
```

Note: Logical records concatenate a string of database records. For example, EMP-JOB-LR contains four record types. Users do not need to know the set relationships between these record types to access information.

CA Culprit accesses logical records as they are encountered in the database. Users can specify logical record key values to selectively retrieve logical records. A logical record key is any logical record field.

Note: For more information about creating and using logical records, see the *CA IDMS Logical Record Facility Guide*.

Table Access and Creation

Non-SQL Defined Tables

CA Culprit users have the ability to access information in **tables** or views of tables as well as create and populate tables. A table maintains information in rows and columns, where one row represents a single record occurrence and one column entry represents a field within the record. Internally, each table is defined as a logical record. The following report illustrates employee information stored in table format.

EMP-NAME		STREET	CITY	STATE	ZIP	PHONE
HERBERT	CRANE	30 HERON AVE	KINGSTON	NJ	21341	2013341433
JANE	FERNDALE	60 FOREST AVE	NEWTON	MA	02576	6178888112
GEORGE	FONRAD	99 VERDE ST	STOUGHTON	MA	02456	6178882323
ROBIN	GARDNER	68 75TH ST	LOWELL	MA	02945	6174521111
DOUGLAS	KAHALLY	56 SPELLING AVE	READING	MA	02317	6173339056
TERENCE	KLWELLEN	12 RAUNCH ST., APT1	DEDHAM	MA	02034	6174456123
SANDY	KRAAMER	56 NASTY WAY	WESTWOOD	MA	02090	6173290002
HERBERT	LIPSICH	6 FORENOON ST	WALDEN	MA	02986	6175555555
NANCY	TERNER	14 TYPE TERR	READING	MA	02317	6173333333

CA Culprit accesses information in a table one row at a time. Users can selectively retrieve rows of the table by coding a SELECT or BYPASS parameter.

Note: For more information about defining and storing tables, see the *CA IDMS ASF User Guide*.

CA Culprit Parameters Unique to Database Access Runs

All of the CA Culprit parameters introduced earlier in this manual are valid for runs that access a CA IDMS/DB database. Four additional parameters, which are listed in the following table, are valid only for database access runs; they are not appropriate for runs that access conventional file structures. In addition, the INPUT and OUTPUT parameters introduced earlier contain some unique options in a CA IDMS/DB environment.

The parameters in the following table are **global** parameters; that is, they are available to every report in the CA Culprit run, unlike report-specific parameters which are available only to the report that defines the parameter.

CA Culprit Parameters	Function
DATABASE	Identifies the dictionary that defines the subschema or table to be accessed
PATH	Defines the database navigation route

CA Culprit Parameters	Function
KEYFILE	Defines the physical characteristics of a file that contains key values with which to access database records directly
KEY	Specifies key values with which to access database records directly
SQL	Introduces an embedded SQL statement to retrieve data from an SQL defined table

Suggested Order

The suggested order of parameters for a non-SQL defined database access run is shown below:

- DATABASE (optional; if specified, it must be first)
- PROFILE
- INPUT, as follows:
  - INPUT with the DB option (required to access database records and logical records)
  - INPUT with the TABLE= option (required to access tables)
- REC (automatically generated by CA Culprit or user-defined)
- PATH (required to access database and logical records; automatically generated for tables)
- SELECT/BYPASS (optional; specifies selection criteria for the PATH parameters)
- Global work field/implied subscript (optional)
- In KEYFILE runs, include the following parameters:
  - KEYFILE (required)
  - REC (optional)
  - SELECT/BYPASS (optional)
- KEY (optional)
- SELECT/BYPASS BUFFER (optional)

- Report-specific parameters (repeat for each report in the CA Culprit run):
  - OUTPUT, as follows:
    - OUTPUT (optional)
    - OUTPUT with the TABLE= keyword expression (required for output to tables)
  - SORT (optional)
  - Title (optional; not appropriate for output to a table)
  - Edit (at least one type 5 edit parameter is required)
  - Process (optional)
  - Work field/implied subscript (optional)

SQL Defined Tables

For more information on querying and manipulating data tables in an SQL defined database, see Accessing SQL Defined Tables (see page 289).

CA Culprit Security Considerations

Security Levels

CA Culprit security is established at several levels:

- **Installation security** establishes whether CA Culprit security is on or off.
- **Product security** establishes whether a user is authorized to use a certain product.
- **User security** establishes whether a user is authorized to access certain products and entities.
- **Passkey** and **row level security** establish whether a user is:
  - Authorized to access a table owned by another user
  - Authorized to access secured rows of a table owned by another user
- **Auto attribute security** establishes whether CA Culprit automatically supplies the characteristics of a file defined to the CA IDMS/DB Integrated Data Dictionary (IDD).

Each type of security is discussed separately below.

Installation Security

CA Culprit is installed with security either on (enabled) or off. Each operating system has an installation parameter that controls this option. Refer to your installation manual for the parameter name. The default is NO. CA Culprit must be reinstalled to change the security option. The Sequential Parameter Listing identifies whether security is in effect with the following statement:

```
INSTALLATION SECURITY OPTION IS NO (or YES)
```

If security is established when CA Culprit is installed, CA Culprit automatically checks the data dictionary to determine the security level in effect and to enforce that security. If installation security is not established, security options set in the data dictionary will be ignored. Installations without a database should install CA Culprit with security off.

Product Security

CA Culprit security is established in the data dictionary with the SECURITY FOR Culprit IS ON/OFF clause of the SET OPTIONS FOR DICTIONARY statement; the user submits this statement to the DDDL compiler. When security in the dictionary is enabled, CA Culprit checks all user authorizations; only authorized users can run CA Culprit jobs that access files or subschemas defined in the data dictionary.

User Security

The ADD USER statement of DDDL syntax documents users in the data dictionary by assigning users the authority to access secured products and to perform secured operations, among many other functions.

The INCLUDE AUTHORITY FOR UPDATE IS Culprit clause specifies that only users with CA Culprit authority can authorize other users to access files and subschemas to run CA Culprit reports. In the example shown below, user ABC can authorize other users access to files and subschemas to run CA Culprit reports, while user DEF is not:

```
ADD USER NAME IS ABC
    INCLUDE AUTHORITY FOR UPDATE IS Culprit.
ADD USER NAME IS DEF
    EXCLUDE AUTHORITY FOR UPDATE IS Culprit.
```

The following clauses give a user access to subschemas, conventional files defined to the data dictionary, and tables:

- **INCLUDE ACCESS TO SUBSCHEMA** authorizes the user to access a specified subschema through CA Culprit when CA Culprit security is enabled.
- **INCLUDE ACCESS TO FILE** authorizes the user to access a specified flat file through CA Culprit when CA Culprit security is enabled.

- **INCLUDE ACCESS TO ASF** gives the user authority to access and create tables.
- **INCLUDE ACCESS TO IDB** gives the user authority to manage the table and to establish communication between CA-ICMS and a personal computer by using IDB (the Information Database).

To remove authority, specify EXCLUDE in place of INCLUDE in the clauses above.

Users can also make changes to record layouts and file definitions if the user is assigned the OVERRIDES option:

```
ADD USER NAME IS ABC
    Culprit OVERRIDES ARE NOT ALLOWED.
```

This clause applies to input files or keyfiles that are defined to the data dictionary. As described in Auto Attribute Security (see page 195), CA Culprit automatically supplies the information necessary to characterize files and fields that are used in a CA Culprit run and defined to the data dictionary. This clause prevents a user from overriding these automatically supplied characteristics, as follows:

- The user cannot define a file in terms of record size, record type, block size, file type, or label type on either KEYFILE or INPUT parameters.
- The user cannot code user-supplied REC parameters for files defined to the data dictionary.

Passkey and Row Level Security

The primary means of controlling access to tables is through the assignment of **passkeys**. A means of providing limited access to tables is through the assignment of **row level security**.

Passkeys give users the authority to perform various operations on tables that they do not own. Row level security allows or prohibits access to secured rows of tables. Individual users are responsible for assigning passkeys and row level security for tables in their private catalogs.

CA Culprit checks passkeys and row level security regardless of whether CA Culprit security is enabled. In a run that consolidates more than one table, CA Culprit applies row level security only to the tables that require it.

Note: For more information about table security considerations, see the *CA IDMS ASF User Guide*.

Auto Attribute Security

Auto attributes define field and file characteristics in IDD that CA Culprit copies from the data dictionary at run time. Auto attributes are established in IDD with the Culprit AUTO ATTRIBUTES ARE ON/OFF clause of the SET OPTIONS FOR DICTIONARY statement.

Auto Attributes Off

When the clause specifies OFF, CA Culprit automatically generates REC parameters for fields defined to the data dictionary; it does not automatically supply the characteristics of input files and keyfiles defined to the data dictionary. For each field that is defined to the data dictionary and coded on a CA Culprit parameter, CA Culprit retrieves the field definition from IDD and generates a corresponding REC parameter. The field can be a field in a database record or a file defined to data dictionary.

Auto Attributes On

When the clause specifies ON, CA Culprit automatically generates REC parameters **and** retrieves the following characteristics of a file defined to the data dictionary:

- Record size
- Block size
- Record type
- File and device type
- VSAM type (for VSAM files only)
- Input user module name (for UM type files only)
- Label type

Example

An example of a file defined to the data dictionary is shown below:

```
ADD
FILE NAME IS EMPLOYEE VERSION IS 788
    DATE CREATED IS mm/dd/yy
    RECORD SIZE IS 80
    BLOCK SIZE IS  400
    RECORDING MODE IS F
    PUBLIC ACCESS IS ALLOWED FOR ALL
    CONTAINS RECORD EMPLOYEE VERSION 788
    CONTAINS RECORD NAME SYNONYM EMPLOYEE VERSION IS 788
    FILE-TYPE IS PS
    DEVICE-TYPE IS 3350.
```

A CA Culprit user can override any of these file characteristics by supplying another value on the INPUT parameter. For example, file EMPLOYEE is defined in the data dictionary with a record size of 80 bytes; a user can override this file characteristic by coding 100 on the INPUT parameter:

```
INPUT 100  FN=EMPLOYEE
```

If auto attributes are in effect (that is, the clause specifies ON), CA Culprit performs the following when it encounters an INPUT or KEYFILE parameter that specifies the FN= keyword expression (which indicates that the file is defined to the data dictionary):

- Retrieves the file characteristics from the data dictionary for all values that are not supplied on the INPUT or KEYFILE parameter.
- Generates a REC parameter for each field defined to the data dictionary and coded on a CA Culprit parameter. It adds 4 bytes to the start position defined in the data dictionary when the following conditions are both true:
  - The file is defined in the DDDL FILE statement with the following substatements:

    ```
    RECORDING MODE V
    RECORD DESCRIPTOR NOT DEFINED
    ```

  - The file is defined in the DDDL FILE statement with any option other than the following substatement:

    ```
    INCLUDE FILE-TYPE IS VS
    ```

Note: For more information about defining products, users, and files to IDD, see the *CA IDMS IDD DDDL Reference Guide*.

DATABASE Parameter

Database Environment

A database environment always consists of at least one data dictionary and one database. A **data dictionary** contains definitions and information about the input data that CA Culprit uses at run time. A **database** contains the input data.

An installation can also establish alternate dictionaries to maintain information specific to a particular application or group of applications; these dictionaries usually share some information with the startup dictionary.

Similarly, company data can be divided among several databases; for example, an employee database, a production database, and a test database. A test database usually contains the same type of information as a production database but contains less data. If the subschemas of the test database and the production database are identical, application programs do not have to modify the name of the subschema to access one or the other database; the programmer merely has to identify the database.

DATABASE Parameter

The DATABASE parameter is an optional parameter that allows users to access a particular data dictionary and to retrieve data from a particular database in a multiple data dictionary/database environment.

Users can also access another DC/UCF system when using the distributed database system. Each DC/UCF system is a **node** in the DDS network. Each node controls one or more data dictionaries or databases.

Note: For more information about distributed database environments, see the *CA IDMS DDS Design and Operations Guide*.

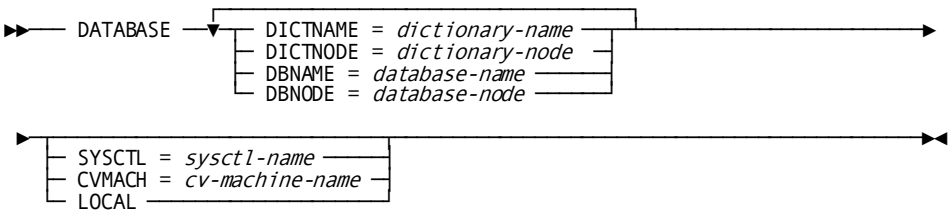
DATABASE Parameter

Purpose

Allows database access.

If specified, DATABASE must be the first parameter in the CA Culprit parameter stream and must be coded starting in column 2; it cannot be part of code copied using USE, =MACRO, or =COPY parameters.

Syntax



Syntax Rules

DICTNAME = *dictionary-name*

Specifies the name of the dictionary to be accessed for a CA Culprit run. The dictionary provides the record and set definitions that describe the subschema to be accessed for the CA Culprit run. The primary dictionary also stores CA Culprit modules that report on the data dictionary.

The default for *dictionary-name* is the start-up (or primary) dictionary. To utilize the default in a CA-ICMS environment, the start-up dictionary must include ASF and IDB.

DICTNODE = *dictionary-node*

(DDS only) Specifies the name of the DC/UCF node that controls the dictionary to be accessed for the CA Culprit run.

DBNAME = *database-name*

Specifies the name of the database from which CA Culprit retrieves data. DBNAME can identify a user database or a secondary data dictionary. This keyword expression is required if the subschema specified on the INPUT parameter maps to more than one database; for example, a test database and a production database.

DBNODE = *database-node*

(DDS only) Specifies the name of the DC/UCF node that controls the named database.

SYSCTL = *sysctl-ddname*

(z/OS and z/VSE only) Specifies a ddname of a SYSCTL file in the JCL stream. The value specified is the default for the CA Culprit run and overrides the default central version in IDMSOPTI. A CA Culprit run that creates or retrieves tables can override the SYSCTL= keyword expression specified on the DATABASE parameter.

CVMACH = *cv-machine-name*

Specifies the name of the virtual machine in which the DC/UCF system is executing; the default is the name specified in an IDMSOPTI module. *Cv-machine-name* is a 2- through 8-character value.

To use this DATABASE parameter option successfully, the CULP0, CULL, and CULE modules must be link edited with an IDMSOPTI module that was assembled with CENTRAL=YES and CVMACH=*cv-machine-name* options.

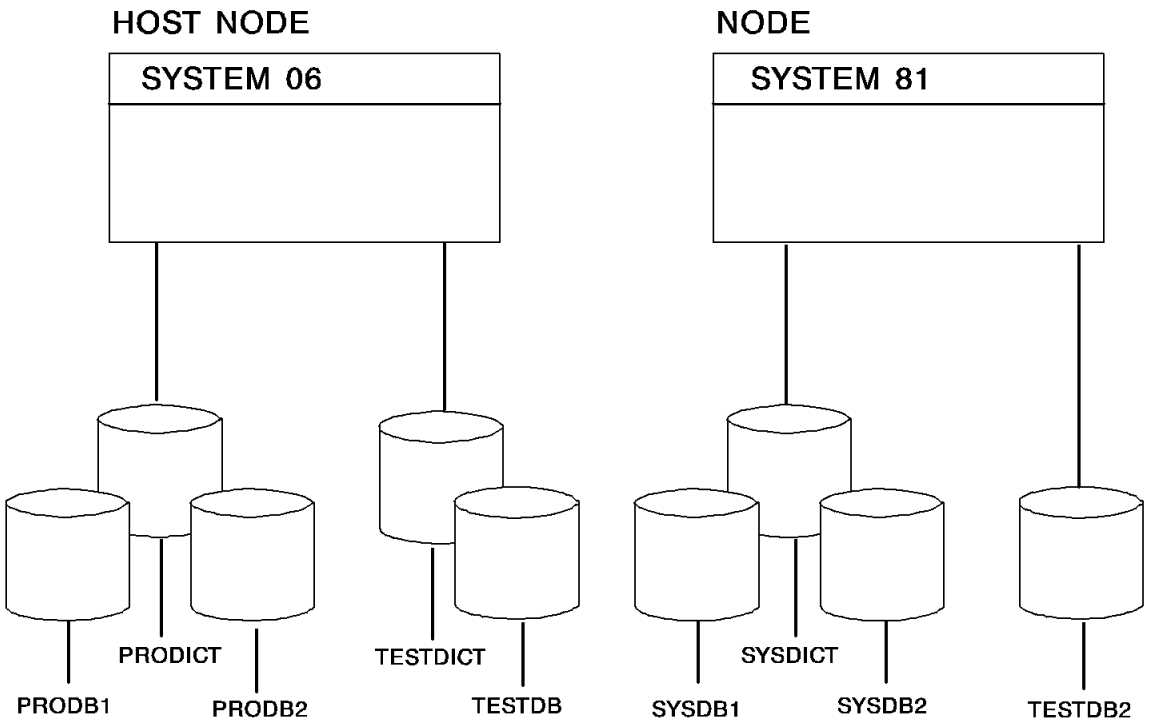
Note: For more information about IDMSOPTI modules, see the *CA IDMS System Operations Guide*.

LOCAL

Specifies that the CA Culprit job is to execute locally.

Examples

The following figure illustrates a multiple dictionary/database DDS environment. The examples of DATABASE parameters that follow reference this figure.



Example 1

```
DATABASE DBNAME=PRODB2
```

The startup dictionary for SYSTEM06 is PRODICT, which contains the schema and subschema source statements that define a view of the database. PRODB2 is a database that supplies the data for the CA Culprit run.

Example 2

```
DATABASE DICTNAME=TESTDICT DBNAME=TESTDB
```

TESTDICT is the name of an alternate dictionary for SYSTEM06; CA Culprit accesses this dictionary for a source description of the subschema specified on the INPUT parameter. TESTDB is the name of an alternate database that supplies the input data for the run.

Example 3

```
DATABASE DICTNAME=PRODICT DBNAME=TESTDICT SYSCTL=SYSTEM06
```

To report on the contents of an alternate dictionary, CA Culprit must access report modules that are stored in the startup dictionary and use the alternate dictionary as input for the CA Culprit job. In this example, PRODICT contains the report modules and TESTDICT is the alternate dictionary. The SYSCTL file with ddname SYSTEM06 is the default for the CA Culprit run.

Example 4

```
DATABASE DBNODE=SYSTEM81
```

The CA Culprit run uses input parameters obtained from the startup dictionary in the host node, SYSTEM06; the startup dictionary is PRODICT. CA Culprit retrieves input data for the run from a database that is controlled by SYSTEM81.

Example 5

```
 DATABASE DICTNODE=SYSTEM81 DBNODE=SYSTEM81
*         DBNAME=TESTDB2
```

DICTNODE= specifies the central version node that controls the dictionary to be accessed for record and set definitions. In this example, the only dictionary defined for node SYSTEM81 is SYSDICT. DBNODE= specifies the central version node that controls the database to be accessed for input data. DBNAME= specifies the name of the database.

More information:

INPUT Parameter (see page 203)
OUTPUT Parameter (see page 270)

PROFILE Parameter

More information:

PROFILE Parameter (see page 29)

PROFILE Parameter with Database Access

In runs that access database records or tables, the PROFILE parameter can supply the ID and password of the user, as well as providing a means to override system defaults, as described in PROFILE Parameter (see page 29).

The PROFILE parameter is required at installations where CA Culprit security is enabled.

More information:

PROFILE Parameter (see page 29)

PROFILE Parameter

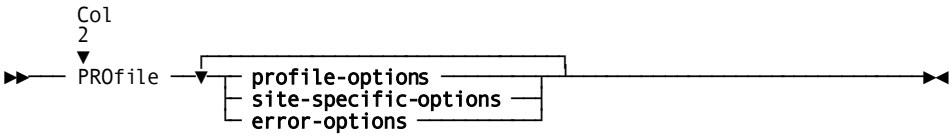
More information:

PROFILE Parameters (see page 32)

Purpose

Override system defaults, and supply the ID and password of the user.

Syntax



Syntax Rules

Except as described below, the syntax rules described in PROFILE Parameter (see page 29), apply to CA Culprit runs in a CA IDMS/DB database environment:

USER = *user-id*

Identifies an authorized user:

- At an installation where CA Culprit security is enabled, the user must be defined to the data dictionary. The ADD USER DDDL statement defines a user to the Integrated Data Dictionary (IDD).

  Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

- To access or create a table, the user must either be the owner of the table or have passkey authorization for the table.

  Note: For more information about passkey authorization, see the *CA IDMS ASF User Guide*.

User-id is a 1- to 32-character alphanumeric value. If *user-id* contains blanks or special characters (that is, other than letters, numbers, or hyphens), it must be enclosed in single quotation marks.

Note: For more information on special coding considerations that apply to a user defined to CA-ICMS, see the *CA IDMS ASF User Guide*.

PW = *password*

Identifies the password associated with the user named on the PROFILE parameter. The password is established in the data dictionary using the PASSWORD IS *password* clause of the ADD USER DDDL statement.

Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

At a CA-ICMS installation, CA-ICMS verifies whether the password is valid and approves or denies access accordingly.

Password is a unique 1- to 8-character value; if special characters are used (that is, characters other than letters, numbers, or hyphens), the password must be enclosed in single quotation marks. Hexadecimal literals must appear in the form X'*password-string*'; for example, PW=X'0102030405'.

The password does not appear on any CA Culprit listings. If no password is associated with the user named on the PROFILE parameter, PW=*password* can be omitted or *password* can be coded as a blank enclosed in single quotation marks (that is, PW=' ').

Usage

CA Culprit establishes the user and password coded on the PROFILE parameter as the default values for the CA Culprit run. Users can override these values in a run that accesses tables by coding user and password information on either the INPUT or OUTPUT parameter.

Examples

Sample PROFILE parameters are shown and described below.

Example 1

```
PROFILE USER=GSR PW=POSSUM
INPUT DB SS=EMPSS01,EMPSCHM,100
```

Before CA Culprit can access records from a database for which security is enabled, CA Culprit passes the user name GSR and password POSSUM to IDD for validation.

Example 2

```
PROFILE USER=EYH PW=RAGTIME
INPUT TABLE=PARTS TYPE=COPY OWNER=LHN LOCATION=ASFDICT
```

To access table PARTS, CA Culprit passes the user's ID and password to IDD for verification. Since the owner of PARTS is EYH, CA Culprit must determine whether user EYH has been issued passkey authorization to access the table.

INPUT Parameter

More information:

INPUT Parameter (see page 43)

When to Use

Coding requirements for an INPUT parameter in a CA IDMS/DB environment depend upon whether the input consists of database records, logical records, or tables:

- For non-SQL defined database and logical records, the INPUT parameter must specify DB and the name of the subschema to be accessed.
- For non-SQL defined tables, the INPUT parameter contains a series of keyword expressions that identify the name, owner, and location of the table, among other things.
- For SQL tables, the input parameter must specify DB(Q), the SQL dictionary, and the name of the SQL schema.

Each source of input is discussed separately below.

Accessing Database and Logical Records

CA Culprit recognizes a run that accesses database or logical records when the INPUT parameter specifies DB and the name of the subschema to be accessed. Syntax specific to database record access is shown below.

INPUT—Accessing Records and Logical Records

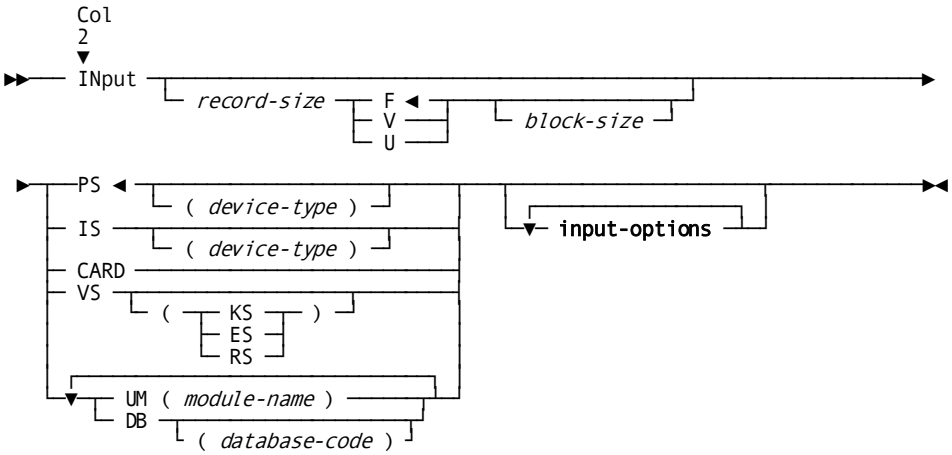
More information:

INPUT Parameter (see page 43)

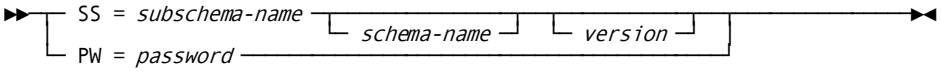
Purpose

Specifies the name of the subschema to be accessed for database records and logical records.

Syntax



Expansion of Input-options



Syntax Rules

Except as described below, the syntax rules presented under INPUT Parameters (see page 43) apply to CA IDMS/DB database record access. Special syntax rules for CA IDMS/DB database record access follow:

record-size

Specifies the number of bytes CA Culprit reserves for the input buffer. *Record-size* is an integer in the range 1 through 32,760; the default is 1000 for a CA IDMS/DB database.

Depending on whether the PATH parameter specifies database or logical records, the minimum size of the input buffer is calculated as follows:

- **For database record access**, the minimum size of the input buffer is the sum of the following items:
  - 38 bytes of input buffer overhead, which includes the values for PATH-ID, REC-NAME, IDMS-STATUS, and LR-STATUS.
  - The total length of all record types on automatic retrieval PATH parameters; each record type is counted the maximum number of times it appears on any single path.
  - The total length of all record types on dummy PATH parameters; a record type is counted each time it appears.
  - Four bytes for each record in the input buffer; this represents the db-key value of each record.
- **For logical record access**, the minimum size of the input buffer is the sum of the following items:
  - 38 bytes of input buffer overhead, as described for database record access above
  - The total length of all record types that define the logical record

UM (*module-name*)

Specifies the name of an input user module. CULLDCLI is the name of the CA Culprit module for access to CA IDMS/DB. CA Culprit automatically supplies this name on the Input Parameter Listing.

DB (*database-code*)

Specifies that CA Culprit is to access database or logical records. *Database-code* is a 1-character alphanumeric value that indicates the database to be accessed, as follows:

Code	What it does
D	(default) Specifies CA IDMS/DB database record access.
I	Specifies IMS (DL/I) database access. **Note:** For more information about accessing an IMS database, see the *CA Culprit IMS Supplement*.
R	Specifies RDMS database access. **Note:** For more information about accessing an RDMS database, see the *CA Culprit RDMS Supplement*.
T	Specifies TIS (formerly called TOTAL) database access. **Note:** For more information about accessing a TIS database, see the *CA Culprit/TOTAL Supplement*.

input-options

Represents coding options that can be specified on an INPUT parameter that accesses database or logical records. See explanation of expanded syntax below.

SS =

Identifies the subschema view to be accessed.

subschema-name

A 1- to 8-character alphanumeric value that identifies the name of the subschema to be accessed. This value is required for database access.

schema-name

A 1- to 8-character value that identifies the schema when *subschema-name* is associated with more than one schema in the data dictionary. This value may be required depending on the schema name option selected at installation; if the SCHMREQ option specifies N (the default), the user can omit the schema name.

Note: For more information about CA Culprit installation options, see the *CA IDMS Installation Guide* for your operating system guide.

version

Identifies the particular version of the schema to be accessed. The default is the highest existing version number for the schema specified.

PW = *password*

Identifies a 1- to 8-character alphanumeric password that may be required by a user input module or a database access module.

If the password contains special characters (that is, anything other than letters, numbers, or hyphens), it must be enclosed in single quotation marks.

The value can also be expressed as a hexadecimal literal; for example, PW=X'0102030405'. *Password* does not appear on any CA Culprit listings.

When CA Culprit accesses database records, it copies the password into the PROGRAM-NAME field of the CA IDMS/DB communications block. Schema-defined database procedures or CA Culprit exit routines can use the PROGRAM-NAME field to validate security and other installation-dependent functions.

Note: For more information about the CA IDMS/DB communications block, see the *CA IDMS DML Reference Guide for COBOL*.

Examples

Sample INPUT parameters for accessing database or logical records are shown and described below.

Example 1

```
INPUT DB SS=EMPSS01,EMPSCHM,100
```

The INPUT parameter identifies CA IDMS/DB database records as the source of input for the CA Culprit run. The string of records returned to the input buffer must have a length less than or equal to 1000 bytes, which is the amount of space CA Culprit allocates for the input buffer by default. Because the subschema view (EMPSS01) is associated with more than one schema, a schema (EMPSCHM) is specified as well as a version number of the schema.

Example 2

```
INPUT 218 DB SS=EMPSS01
PATHAA DEPARTMENT EMPLOYEE
```

CA Culprit accesses database records that are defined in the EMPSS01 subschema. The size specified for the input buffer is the sum of the following items:

- The length of the input buffer overhead (38 bytes)
- The length of the DEPARTMENT record (56 bytes)
- The length of the EMPLOYEE record (116 bytes)
- The length of the db-key field associated with each record type specified on the PATH parameter (4 bytes times 2)

Example 3

```
IN 210 DB SS=EMPLR
PATHBB DEPT-EMPLOYEE-LR
```

In this example of the INPUT DB parameter, the input to the CA Culprit run is the DEPT-EMPLOYEE-LR logical record. The size specified for the input buffer is the sum of the following items:

- The length of the input buffer overhead (38 bytes)
- The sum of the lengths of the records that define the logical record; in this example, the logical record contains the DEPARTMENT and EMPLOYEE records defined in the example above

Accessing Tables

To access tables in a non-SQL defined database, the INPUT parameter identifies the name and owner of a table or a view of a table, as well as other options associated with the table.

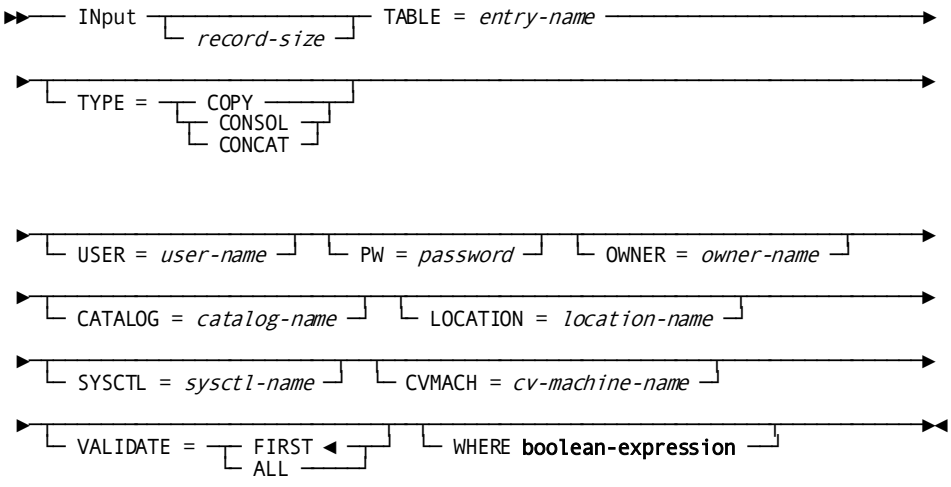
CA Culprit can also consolidate tables stored in different dictionaries or on different systems. For example, CA Culprit can access inventory information stored in tables in different cities. Up to 999 tables can be consolidated in a CA Culprit run. Each table is processed in the order it appears on the INPUT parameters. CA Culprit assigns each table an ID starting with 001 for the primary table.

INPUT—Accessing Tables

Purpose

Identifies the name and owner of a table or a view of a table, as well as other options associated with the table.

Syntax



Syntax Rules

INPUT

Specifies the type of parameter. It must start in column 2.

In a CA Culprit run that consolidates tables, the table named on the first INPUT parameter is the **primary** table. CA Culprit uses the primary table to:

- Generate the PATH parameter
- Generate a REC parameter for each column referenced in the table

Tables named on subsequent INPUT parameters are **secondary** tables. The column definitions of each secondary table must be identical to those of the primary table.

record-size

Specifies the row length of the table. It must be an integer in the range 1 through 32,760. The following considerations apply:

- If omitted, *record-size* defaults to the size of the largest table to be consolidated.
- If you provide too small a value, CA Culprit automatically overrides the value.
- If you provide a value that exceeds CA Culprit's requirements, the extra space is available for dummy buffer fields.

TABLE = *entry-name*

Specifies the name of a table. TABLE = *must* be the first keyword coded on the INPUT card. *Entry-name* is a 1- to 64-character alphanumeric expression. If the name contains embedded blanks or special characters such as punctuation marks or symbols, it must be enclosed in single quotation marks. The following rules apply:

- When enclosed in quotation marks, the entire name must be on the same line:

```
IN   TABLE='MZC.REPTS85.BUDGET'
```

- When not enclosed in quotation marks, the name can span multiple lines and comments can appear on any line:

```
 IN   TABLE=MZC.          $ Owner.Folder.Object
*           REPTS85       $ Note that the period
*           .BUDGET       $ can appear on either line
```

TYPE = COPY/CONSOL

Specifies the primary and secondary tables to be consolidated:

- COPY identifies the primary table.
- CONSOL (CONCAT) identifies a secondary table.

USER = *user-name*

Identifies the individual with authority to sign on to the catalog and overrides the user specified on the PROFILE parameter, if any. *User-name* is a 1- to 32-character value.

PW = *password*

Is the user's 1- to 8-character alphanumeric password. It overrides the password specified on the PROFILE parameter.

OWNER = *owner-name*

Identifies the table owner. When the user is not the table owner, this keyword expression is required. CA Culprit determines whether the owner has assigned the user passkey authority to read the table.

CATALOG = *catalog-name*

Identifies the catalog (dictionary) containing the table definition. This keyword allows you to specify a different catalog than the one given on the DICTNAME= clause of the DATABASE parameter. Consequently, you can consolidate data tables from any number of catalogs (dictionaries).

Catalog-name is a 1- to 8-character alphanumeric expression. If specified, *catalog-name* overrides the dictionary specified on the DATABASE parameter. If *catalog-name* is not specified, the default is as follows:

- CA Culprit uses the database name (if specified) in the DBNAME= keyword expression on the DATABASE parameter.
- If no database name is specified, CA Culprit uses the data dictionary name (if specified) in the DICTNAME= keyword expression on the DATABASE parameter.
- If neither a database or data dictionary name is specified on the DATABASE parameter, the default is the startup dictionary.

LOCATION = *location-name*

(DDS users only) Identifies the DDS node used to access the catalog containing the table. This keyword expression allows you to consolidate data tables from any number of nodes in a given DDS network.

Location-name is a 1- to 8-character value. The default is the value specified in the DICTNODE= keyword expression on the DATABASE parameter.

SYSCTL = *sysctl-name*

(z/OS and z/VSE users only) Overrides the SYSCTL= clause on the DATABASE parameter. If there is no sysctl value on the DATABASE parameter, CA Culprit uses the default value in IDMSOPTI. The SYSCTL= keyword expression allows users to consolidate tables from any number of DC/UCF systems.

CVMACH = *cv-machine-name*

(z/VM users only) Overrides the value of the CVMACH= clause on the DATABASE parameter and the central version specified in an IDMSOPTI module. *Cv-machine-name* is a 2- through 8-character value that identifies the virtual machine in which the IDMS-CV system is executing. With this keyword expression, CA Culprit can consolidate tables from any number of central versions in the same CA Culprit job step.

VALIDATE =

Compares either the first column or all columns of the secondary tables to the columns in the primary table:

FIRST

Validates the first column in every secondary table to the first column in the primary table. CA Culprit compares the columns' offset, size, usage mode, data type, and decimal point.

FIRST is the default.

ALL

Validates the compatibility of every column in the secondary tables to every column in the primary table.

WHERE boolean-expression

Applies WHERE clause criteria to all tables being consolidated. The WHERE criteria, if specified, must appear after all keyword expressions coded for the primary table (that is, the first INPUT parameter).

For syntax rules that apply to the WHERE clause, see PATH Parameter (see page 220).

For the expanded syntax for **boolean-expression**, see SELECT / BYPASS — Overview (see page 59).

Usage

CA Culprit produces a validation report in the Input Parameter Listing, as shown in the following figure. The report indicates column definitions in secondary tables that do not match the definitions of the primary table. If FIRST is the option specified, only the definition of the first column appears in the report; if ALL is specified, any of the column definitions can appear in the report.

```
VALIDATE   COLUMN    TABLE-ID     COLUMN                    POSITION IN   INTERNAL   USAGE       DATA     DECIMAL
FIELDS     NUMBER                   NAME                    DATA RECORD    SIZE     MODE       TYPE     POINTS
********
             001       001       DEPT-NAME-0410               00001       0045     DISPLAY     TEXT
             001       002       DEPT-ID-0410                 00001       0005     DISPLAY     TEXT
 E C220002 FIELD INCOMPATIBLE WITH PRIMARY TABLE                          ****
             002       001       EMP-ID-0415                  00046       0006     DISPLAY     TEXT
             002       002       DEPT-NAME-0410               00006       0045     DISPLAY     TEXT
 E C220002 FIELD INCOMPATIBLE WITH PRIMARY TABLE              *****       ****
             003       001       EMP-NAME-0415                00052       0025     DISPLAY     TEXT
             003       002       EMP-ID-0415                  00051       0006     DISPLAY     TEXT
 E C220002 FIELD INCOMPATIBLE WITH PRIMARY TABLE              *****       ****
             004       002       EMP-NAME-0415                00057       0025     DISPLAY     TEXT
 E C220001 NO CORRESPONDING FIELD ON PRIMARY TABLE
             005       002       SALARY-AMOUNT-0420           00082       0014     DISPLAY     TEXT
 E C220001 NO CORRESPONDING FIELD ON PRIMARY TABLE
```

Examples

Example 1: Consolidating Three Data Tables

```
 DATABASE DICTNAME=ASFDICT
 PROFILE USER=DJM PW=WALNUT
 INPUT TABLE=ACCOUNTING TYPE=COPY VALIDATE=ALL
*      WHERE EMP-NAME-0415 LE 'M'
 INPUT TABLE=PERSONNEL TYPE=CONSOL
 INPUT TABLE='PUBLIC RELATIONS' TYPE=CONSOL
```

In this example, CA Culprit consolidates tables for the Accounting, Personnel, and Public Relations departments. The DATABASE and PROFILE parameters provide the name of the catalog that contains the table definitions, the user's ID and the user's password. At run time, CA Culprit also determines that the user is also the owner of the tables.

The ACCOUNTING table is the primary table. The VALIDATE= keyword expression instructs CA Culprit to compare the column definitions in each of the secondary tables to the column definitions in the primary table. The WHERE clause selects only those rows in each table where the name of the employee begins with a letter less than or equal to 'M'.

Example 2: Using Multiple Dictionaries

```
 IN    TABLE=ACCOUNTING TYPE=COPY USER=LSB PW=PECAN
*      CATALOG=ASFDICT      SYSCTL=SYSTEM85
 IN    TABLE='INTERNAL SOFTWARE'  TYPE=CONSOL
*      USER=LSB    PW=PECAN OWNER=MCK
*      CATALOG=TESTDICT     SYSCTL=SYSTEM91
```

In this example, ACCOUNTING resides in dictionary ASFDICT in a DC/UCF system called SYSTEM85. INTERNAL SOFTWARE resides in the TESTDICT dictionary on SYSTEM91. The SYSCTL= keyword values refer to ddnames that appear in a z/OS or z/VSE JCL stream; in z/OS, for example:

```
//SYSTEM85  DD  DSN=DBDC.SYSTEM85.SYSCTL,DISP=SHR
//SYSTEM91  DD  DSN=DBDC.SYSTEM91.SYSCTL,DISP=SHR
```

REC Parameter

More information:

REC Parameter—Overview (see page 51)

Two Types

REC parameters define input field characteristics such as field length, field position, and data type. REC parameters are either automatically generated by CA Culprit or coded by the user. Each type of REC parameter is discussed separately below.

Automatically Generated REC Parameters

For database records, logical records, files defined to the data dictionary, and tables CA Culprit automatically supplies a REC parameter for each input field referenced on another CA Culprit parameter; for example, if an edit parameter references DEPT-ID-0410, CA Culprit provides the REC parameter that defines the field.

REC cards are also automatically generated for SQL defined tables. For information on using CA Culprit with an SQL defined database, see Accessing SQL Defined Tables (see page 289).

Considerations

The following considerations apply to CA Culprit generated REC parameters:

- Fields defined to the data dictionary can have auto-header literals associated with the data dictionary definition. To define an auto-header literal to the data dictionary, use the Record Element DDDL substatement as shown in the example below.

  Note: For more information, see the *CA IDMS IDD DDDL Reference Guide*.

  ```
  RECORD ELEMENT IS NAME VERSION 788
  LINE IS 000100
  LEVEL NUMBER IS 02
  USAGE IS DISPLAY
  Culprit HEADER
      'EMPLOYEE'
      'NAME'.
  ```

- For multiply-occurring fields, CA Culprit assigns each field a numeric group id, starting sequentially with 00. If more than 100 groups are generated, CA Culprit assigns lowercase alphabetic group ids, starting with aa. User-supplied group definitions should specify uppercase alphabetic group ids to distinguish them from generated group definitions.
- CA Culprit converts numeric single or double precision floating point input fields into 4-byte alphanumeric fields. The CA Culprit module CULLUS36 (Floating Point Conversion) converts these alphanumeric fields into 16-byte packed signed decimal integers.

The following report illustrates CA Culprit generated REC parameters for a database record that contains multiply-occurring fields; the generated REC parameters appear on the Input Parameter Listing.

```
********
REC      START SIZE TYPE DP FIELD-NAME                    RECORD-NAME,LEVEL
********
 REC     00169  001         GENERATED-GROUP-00             GROUP 00       00080.00010         $$ GENERATED
 REC     00169  001         GENERATED-GROUP-01             GROUP 01       00080.00010         $$ GENERATED
 REC     00039  006         CLAIM-DATE-0405                DENTAL-CLAIM   $$ GENERATED
 REC     00164  002   1     NUMBER-OF-PROCEDURES-0405      DENTAL-CLAIM   $$ GENERATED
 REC     00045  025         PATIENT-NAME-0405              DENTAL-CLAIM   $$ GENERATED
 REC     00009  004   2     PROCEDURE-CODE-0405            DENTAL-CLAIM   ELMNT 01  $$ GENERATED
 REC     00003  006         SERVICE-DATE-0405              DENTAL-CLAIM   ELMNT 00  $$ GENERATED
mm/dd/yy                      RUN TIME MESSAGES               PAGE     1
```

Note: CA Culprit Generated REC Parameters: For each field referenced on other CA Culprit parameters, CA Culprit supplies the start position of the field in the input buffer, the field length, field name, data type, decimal positions, and database record name. For each field that occurs a multiple number of times, CA Culprit generates a GROUP REC parameter and an ELMNT REC parameter.
In this example, the multiply-occurring group is 80 bytes long and repeats ten times.

User-supplied REC Parameters

When CA Culprit accesses database records, logical records, or tables, user-supplied REC parameters are coded for the following reasons:

- To define fields contained in a key file.
- To rename fields defined to the data dictionary.
- To redefine fields defined to the data dictionary; for example, to redefine an alphanumeric date field as a numeric field.
- To define a portion of a field; for example, the month portion of a date field.

Associated headers can be included on user-supplied field definitions because the headers are not generated when a REC parameter overrides a data dictionary field definition.

More information:

KEY and KEYFILE Parameters (see page 251)

REC Parameter

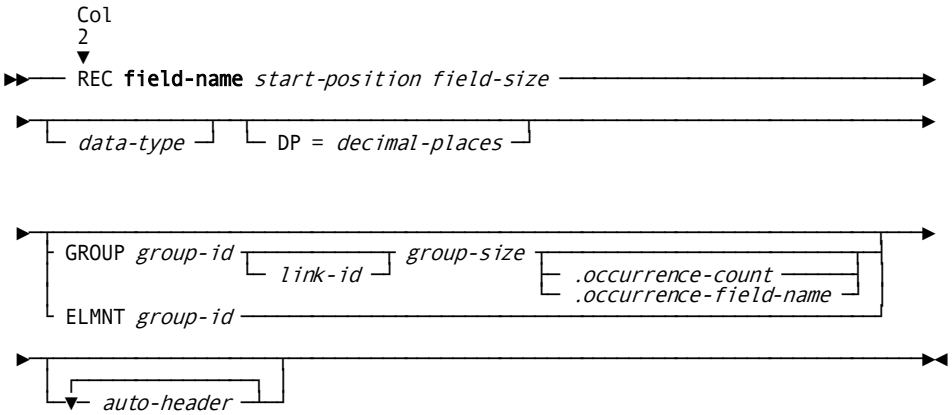
More information:

KEY and KEYFILE Parameters (see page 251)
REC Parameter—Overview (see page 51)

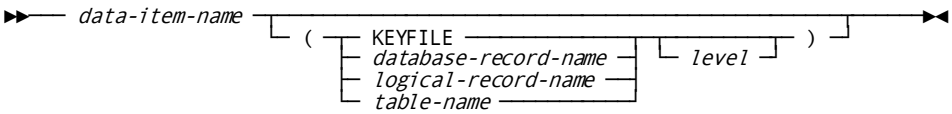
Purpose

Defines input field characteristics.

Syntax



Expansion of Field-name



Syntax Rules

Syntax rules for database access, key files, or files defined to the data dictionary are as described under REC Parameter (see page 52), except for those items explained below:

field-name

Specifies a name of a field in a database record, table, key file record, logical record, or record in a file defined to the data dictionary. See expansion of **field-name** below.

data-item-name

A 1- to 32-character name that identifies the field. It can be coded in any column after REC. The name can consist of letters, numbers, or hyphens. The following coding considerations apply to *data-item-name*:

- At least one alphabetic character is required.
- The name cannot begin or end with a hyphen.
- The name cannot be any of the reserved words listed in Reserved Words (see page 369).

KEYFILE

A keyword that must be specified if *data-item-name* resides in a key file.

database-record-name

Specifies the name of the database record associated with the field; the database record must be specified on the PATH parameter.

logical-record-name

Specifies the name of the logical record associated with the field; the name must be specified on the PATH parameter.

table-name

Specifies the table named on the first (or only) INPUT parameter.

level

Specifies the relative occurrence level of the database record or logical record within the CA Culprit input buffer. The value is an integer in the range 1 through 256; the default is 1. If specified, *level* must be separated from the name of the database record or logical record by a comma or a space and must precede a closing parenthesis.

Multiple levels can occur in a database access run under the following circumstances:

- The record name appears more than once on a single path, as shown in the bill-of-materials example below:

```
 PATH01 EMPLOYEE STRUCTURE(MANAGES)
*       EMPLOYEE(REPORTS-TO)
```

- The record name appears on a dummy path as well as on an automatic retrieval path, as discussed in PATH Parameter (see page 220).

The first occurrence of a repeating database record on an automatic retrieval PATH parameter is assigned level 1; the second occurrence is assigned level 2, and so on. If the record appears on one or more dummy PATH parameters, the first occurrence on the dummy path is one higher than the highest level for that record on the automatic retrieval path; the second occurrence is one higher than the previous appearance, and so on.

For multiply-occurring fields, *level* must be specified on the REC parameter that specifies the keyword GROUP as well as on the REC parameter that specifies the keyword ELMNT.

start-position

Specifies the start position of the field relative to the beginning of a record or the input buffer; a field that is not qualified by a record name, table name, or KEYFILE refers to a field in a position relative to the beginning of the input buffer.

The Input Parameter Listing indicates the start position of the field in the input buffer. Database and logical records begin in byte position 39 in the input buffer; tables begin in byte position 95, provided the table is not the result of a JOIN operation.

The user-supplied start position is 38 bytes less than the value CA Culprit reports on the Input Parameter Listing (38 bytes represents the amount of input buffer overhead). For example, from the generated REC parameters shown in the screen capture at the end of the Automatically Generated REC Parameters (see page 213) section, the user-supplied start position for PATIENT-NAME-0405 is 7.

Usage

If one of the options KEYFILE/*database-record-name/ logical-record-name/table-name* is specified, it must be enclosed in parentheses.

If *data-item-name* is a multiply-occurring field, both the REC parameter that specifies the keyword GROUP and the REC parameter that specifies the keyword ELMNT must qualify *data-item-name* with one of these options.

Examples

Sample user-supplied REC parameters are shown and described below.

Example 1: KEYFILE-Defined Field

```
REC KEY-EMP-NAME (KEYFILE)  10 25 'EMPLOYEE NAME'
```

KEY-EMP-NAME describes a field in a key file record. The 25-byte alphanumeric field starts in byte 10 of the key file record. EMPLOYEE NAME appears as a column heading if the field is referenced on a type 5 edit parameter that specifies header origin code HR.

Example 2: Database Record-Defined Field

```
REC EMP-ID        (EMPLOYEE,1)   1 4 2  'EMPLOYEE ID'
REC EMP-NAME      (EMPLOYEE,2)   5 25   'EMPLOYEE NAME'

PATHAA EMPLOYEE STRUCTURE(MANAGES) EMPLOYEE(REPORTS-TO)
```

These REC parameters rename the EMP-ID-0415 and EMP-NAME-0415 fields in the EMPLOYEE database record, which appears in Employee Database Subschema (see page 415). EMP-ID is a field in the first occurrence of the EMPLOYEE database record on the PATH parameter; this 4-byte zoned decimal field begins in position 1 of the EMPLOYEE database record. EMP-NAME is a field in the second occurrence of the EMPLOYEE database record, qualified by (REPORTS TO) on the PATH parameter. This 25-byte alphanumeric field begins in position 5 of the EMPLOYEE database record.

Example 3: Database Record-Defined Field

```
 REC NUMBER-OF-PROCEDURES (DENTAL-CLAIM)   126  2  1
 REC CHARGES (DENTAL-CLAIM)     131 GROUP AA
*                               80.NUMBER-OF-PROCEDURES (DENTAL-CLAIM)
 REC TOOTH-NUMBER (DENTAL-CLAIM)   1 2 2 ELMNT AA
 REC SERVICE-DATE (DENTAL-CLAIM)   3 6 2 ELMNT AA
 REC PROCEDURE-CODE (DENTAL-CLAIM) 9 4 2 ELMNT AA
```

The fields defined to the data dictionary for the DENTAL-CLAIM record specify -0405 to identify the record. The user-supplied REC parameters in this example alter the data dictionary-defined fields:

- Remove the -0405 subscript from all the field names
- Redefine SERVICE-DATE as a zoned decimal rather than an alphanumeric field

The REC parameters define multiply-occurring fields that repeat a variable number of times depending on the value for NUMBER-OF-PROCEDURES.
NUMBER-OF-PROCEDURES begins in byte position 126 of the DENTAL-CLAIM record.
The REC parameter that specifies the keyword GROUP identifies byte position 131 as the start position of the multiply-occurring fields in the database record. The total length of the multiply-occurring fields associated with GROUP AA is 80 bytes. The three REC parameters that specify the keyword ELMNT identify elements of GROUP AA that occur in the first 12 bytes of the multiply-occurring field.

Example 4: Database Record-Defined Field

```
 REC NUMBER-OF-PROCEDURES-2 (DENTAL-CLAIM,2)   126  2  1
 REC CHARGES-2 (DENTAL-CLAIM,2)  131 GROUP AA
*                          80.NUMBER-OF-PROCEDURES (DENTAL-CLAIM,2)
 REC TOOTH-NUMBER-2 (DENTAL-CLAIM,2)   1 2 2 ELMNT AA
 REC SERVICE-DATE-2 (DENTAL-CLAIM,2)   3 6 2 ELMNT AA
 REC PROCEDURE-CODE-2 (DENTAL-CLAIM,2) 9 4 2 ELMNT AA
```

This example is the same as Example 3, except that each REC parameter specifies a level number. The level number indicates that the REC parameters define fields in the second occurrence of the DENTAL-CLAIM record in the input buffer. For multiply-occurring field definitions, the occurrence level must be specified for both the REC parameter that defines the group and the REC parameters that define the elements of the group.

Example 5: Logical Record-Defined Field

```
REC DEPT-ID (EMP-DEPT-LR) 49  4    'DEPARTMENT' 'ID'
```

DEPT-ID-0410 is a field in the EMP-DEPT-LR. The user-supplied REC parameter renames the field and redefines it as an alphanumeric value. The field begins in byte position 49 of the logical record; it is processed as a 4-byte alphanumeric field. DEPARTMENT ID can appear as a 2-line column header if the field is referenced on a type 5 edit parameter that specifies header origin code HR.

Example 6: Table-Defined Field

```
INPUT TABLE=ACCOUNTING TYPE=COPY
REC DEPT-NAME (ACCOUNTING)   57  25
```

DEPT-NAME-0410 is the first field in table ACCOUNTING, which the user-defined REC parameter renames DEPT-NAME. The alphanumeric field begins in byte position 57 of the table row for a length of 25 bytes. The CA Culprit supplied REC parameter appears in the Input Parameter Listing as follows:

```
********
REC      START  SIZE  TYPE  DP    FIELD-NAME     RECORD-NAME,LEVEL
********
 REC     00095  00025        DEPT-NAME          ACCOUNTING
```

Note: The generated start position reflects 38 bytes of overhead at the beginning of the input buffer.

PATH Parameter

What It Does

A PATH parameter defines a path through the database. CA Culprit navigates the path and returns information to the input buffer to be processed. The following formats define a path:

- A list of one or more database records that can be followed by a logical record
- A single logical record

At least one PATH parameter is required for database access. CA Culprit automatically generates a PATH parameter for runs that access one or more ASF tables.

Considerations that apply to accessing database records, logical records, and tables and filling the input buffer are discussed separately below, followed by the syntax for the PATH parameter.

Accessing Database Records

When accessing records in a non-SQL defined database, CA Culprit follows the route the user defines in the PATH parameter. The PATH parameter provides CA Culprit with the following information:

- An entry point into the database
- An order in which to retrieve information
- Identifiers that differentiate one path from another

Each item is discussed separately below. All of the example PATH parameters that appear in each discussion refer to Employee Database Subschema (see page 415); the figure illustrates the structure of a database designed to manage employee-related data.

Establishing an Entry Point

An **entry record** identifies a starting point from which to navigate the database. The record is the first record named on the PATH parameter. Each PATH parameter specified for the run must have the same entry record. In the example shown below, the EMPLOYEE record is the entry record for each PATH parameter; CA Culprit accesses an EMPLOYEE record and navigates path AA; it then navigates paths BB and CC before accessing the next occurrence of the EMPLOYEE record:

```
PATHAA EMPLOYEE DEPARTMENT
PATHBB EMPLOYEE COVERAGE DENTAL-CLAIM
PATHCC EMPLOYEE EMPOSITION JOB
```

CA Culprit accesses the entry record by using either an area sweep (that is, reading each record in the area) or by using the CALC-key, index-key, or db-key value of a field in the entry record. CA Culprit obtains these key values from either a KEY parameter or a KEYFILE parameter.

Defining a Path

A **set relationship** must exist between the records specified on the path, unless the PATH parameter specifies a dummy path; dummy paths are discussed later in this chapter. The sets can be implemented with embedded pointers or with an index. The following rules apply to the sequence of records that define the path:

- The relationship between the entry record and the next record on the path can be either owner-to-member or member-to-owner, as shown in the example below:

  ```
  Correct Paths:     EMPLOYEE EMPOSITION
                     EMPLOYEE DEPARTMENT

  Incorrect Path:    EMPLOYEE JOB
  ```

- The record that follows an owner-to-member traversal must be related to the member record, as shown below:

  ```
  Correct Path:    EMPLOYEE EMPOSITION JOB

  Incorrect Path:  EMPLOYEE EMPOSITION EXPERTISE
  ```

- The record that follows a member-to-owner traversal can be related to the owner record or to any previous record named on the path, provided the previous record and any intervening records were retrieved using a member-to-owner traversal, as shown below:

  ```
  Correct Paths:   COVERAGE EMPLOYEE OFFICE EXPERTISE
                   EXPERTISE EMPLOYEE OFFICE SKILL
                   EXPERTISE EMPLOYEE DEPARTMENT COVERAGE

  Incorrect Path:  EXPERTISE EMPLOYEE EMPOSITION SKILL
  ```

- A single record type can occur more than once on a path, as shown below:

  ```
  Correct Paths:  EMPLOYEE STRUCTURE (REPORTS-TO) EMPLOYEE (MANAGES)
                  EMPLOYEE COVERAGE EMPLOYEE EXPERTISE
  ```

Specifying Path Identifiers

Path identifiers are unique codes that make it possible for CA Culprit to perform the following functions:

- Navigate an incomplete database path
- Navigate more than one database path

Each time CA Culprit navigates the database, it places the path ID value in the PATH-ID field in the input buffer; users can test the value of PATH-ID to determine which path CA Culprit was able to navigate.

Path identifiers can be of several types:

Primary Path

A primary path ID indicates that CA Culprit was able to access an occurrence of each record type specified on the PATH parameter; that is, it was able to navigate the path completely.

If more than one PATH parameter is specified, CA Culprit navigates the paths as follows:

- The first path that CA Culprit navigates is the path with the lowest sort-sequenced primary path id; for example, CA Culprit navigates PATHAA before PATHBB.
- The next path that CA Culprit navigates is the path that contains the most similar sequence of record types. In the example shown below, CA Culprit navigates PATHCC before PATHBB because PATHCC has more records in common with PATHAA:

```
PATHAA DEPARTMENT EMPLOYEE COVERAGE
PATHBB DEPARTMENT
PATHCC DEPARTMENT EMPLOYEE EMPOSITION JOB
```

Alternate Path

An alternate path identifier indicates that CA Culprit is to return a partial string to the input buffer if it is not able to build a complete string. For example, because the PATH parameter shown below specifies path ID CC, CA Culprit can return a string to the input buffer that contains occurrences of the DEPARTMENT and EMPLOYEE records if an occurrence of the COVERAGE record does not exist or was not selected:

```
PATHAA DEPARTMENT EMPLOYEE COVERAGE (CC)
```

Null Path

A null path identifier is a special type of alternate path id; it indicates that an occurrence of the entry record was either not encountered or not selected for processing. A null path ID follows the entry record on the PATH parameter, as shown below:

```
PATH01 EMPLOYEE (02) EXPERTISE SKILL
```

Dummy Path

A dummy path identifier allows CA Culprit to access database records or logical records directly within the procedure code by using the DB-EXIT facility. The identifier for a dummy PATH parameter is two hyphens (--), as shown below:

```
PATH-- EMPLOYEE JOB DEPARTMENT
```

The above example indicates that the records specified on a dummy path do not have to participate in set relationships with one another because the user controls the record retrieval process.

Path Statistics

The following report shows the path statistics CA Culprit lists on the Run Time Messages report. The listing indicates how many times CA Culprit was able to navigate each path and how many occurrences of each database record CA Culprit accessed.

```
mm/dd/yy          RUN TIME MESSAGES                PAGE     1

  IDMS DATABASE EXTRACT STATISTICS
  STRINGS RETURNED FOR PATH AA -          34
  STRINGS RETURNED FOR PATH BB -           0
  STRINGS RETURNED FOR PATH CC -           0
  STRINGS RETURNED FOR PATH DD -           1
  STRINGS RETURNED FOR PATH EE -          29
  STRINGS RETURNED FOR PATH FF -           0
  STRINGS RETURNED FOR PATH GG -           0

  RECORD NAME       NUMBER READ
 DEPARTMENT                  4
 EMPLOYEE                   29
 EMPOSITION                 34
 OFFICE                     29
```

Note: CA Culprit navigated path AA 34 times, path DD 1 time, and path EE 29 times. In order to complete these paths, CA Culprit retrieved 4 DEPARTMENT record occurrences, 29 EMPLOYEE record occurrences, 34 EMPOSITION record occurrences, and 29 OFFICE record occurrences.

More information:

Database Field Name References (see page 244)
The DB-EXIT Facility (see page 429)

Accessing Logical Records

Logical records allow users to access database records without knowing the structure of the database. A logical record is a concatenation of two or more database records. Typically, the PATH parameter specifies one logical record; however, a PATH parameter may also specify a string of database records followed by the logical record, as shown in the examples below:

```
PATHYZ  EMP-JOB-LR
PATHCF  DEPARTMENT EMP-JOB-LR
```

A logical record PATH parameter can specify a null path identifier and a WHERE clause. CA Culprit returns the null path ID to the PATH-ID field if it was not able to retrieve an occurrence of the logical record. The WHERE clause allows CA Culprit to selectively retrieve logical record occurrences; in the example shown below, the WHERE clause selects logical record occurrences that specify 5100 for the department id:

```
PATHDD EMP-JOB-LR WHERE DEPT-ID-0410 EQ '5100'
```

Accessing non-SQL Defined Tables

CA Culprit automatically generates a PATH parameter for runs that access one or more tables in a non-SQL defined database. The PATH parameter names the primary input table. CA Culprit retrieves information from the table or tables one row at a time. Users can select rows to be processed by specifying:

- A WHERE clause on the INPUT parameter defined for the primary table
- SELECT/BYPASS parameters
- Type 7 logic for individual reports

Filling the Input Buffer

The example shown below illustrates three primary paths:

```
PATHAA DEPARTMENT EMPLOYEE EXPERTISE
PATHBB DEPARTMENT
PATHCC DEPARTMENT EMPLOYEE
```

At run time, CA Culprit accesses and processes the records specified on these PATH parameters in the following sequence:

1. CA Culprit starts with the path that has the lowest sort-sequenced path id, and then builds a string of database records; in this example, CA Culprit navigates path AA first.
2. CA Culprit returns the string to the input buffer, which is processed by each report in the CA Culprit run.

3. CA Culprit obtains a new occurrence of the last record type retrieved in the path (EXPERTISE) and returns the new string to the input buffer. When no further occurrences of EXPERTISE exist for a given employee, CA Culprit navigates path CC, which specifies the same DEPARTMENT EMPLOYEE record sequence as path AA.
4. CA Culprit obtains a new occurrence of the next-to-last record on path AA (EMPLOYEE) and extends the string again. When no further occurrences of EMPLOYEE exist for a given department, CA Culprit navigates path BB.

The iteration process continues until all possible occurrences of each record type on the paths have been returned.

CA Culprit does not clear the input buffer before it builds the next string of database records. Therefore, if the current string is shorter than the previous string, residual information will exist in the input buffer. The user must test the value of PATH-ID in the input buffer to determine which path CA Culprit has navigated.

The following figure lists a subset of information stored in occurrences of the EMPLOYEE, EXPERTISE, and SKILL records from the employee database in Employee Database Subschema (see page 415).

EMPLOYEE NAME	EXPERTISE LEVEL	SKILL
KATHERINE O'HEARN	03	FORTRAN
		COBOL
		ASSEMBLER
	02	IDD
LAURA PENMEN	-	-
LARRY LITERATA	03	EDITING
	02	INTERACT
		COBOL

Examples

The following examples indicate the contents of the input buffer when different paths are specified; residual information in the input buffer is enclosed in parentheses.

Example 1: Primary Path Id

```
PATHAA EMPLOYEE EXPERTISE SKILL
```

Since the path specifies only a primary path id, CA Culprit must retrieve an occurrence of each database record specified; therefore, CA Culprit will not retrieve the EMPLOYEE record occurrence for LAURA PENMEN since there are no occurrences of EXPERTISE and SKILL records for this employee. The contents of each input buffer appear below:

```
PATH ID      EMPLOYEE NAME              EXPERTISE     SKILL
                                          LEVEL

   AA        KATHERINE O'HEARN             03        FORTRAN
   AA        KATHERINE O'HEARN             03        COBOL
   AA        KATHERINE O'HEARN             03        ASSEMBLER
   AA        KATHERINE O'HEARN             02        IDD
   AA        LARRY     LITERATA            03        EDITING
   AA        LARRY     LITERATA            02        INTERACT
   AA        LARRY     LITERATA            02        COBOL
```

Example 2: Alternate Path Id

```
PATHAA EMPLOYEE EXPERTISE (BB) SKILL
```

Since the PATH parameter specifies an alternate path id, CA Culprit can process an incomplete path. In this example, CA Culprit can retrieve occurrences of EMPLOYEE records that do not have an associated occurrence of an EXPERTISE record. The contents of the input buffer shown below indicate that CA Culprit was able to retrieve the LAURA PENMEN record occurrence. Note however that KATHERINE O'HEARN's previously retrieved expertise level (02) and skill (IDD) remain in the PENMEN buffer.

```
PATH ID      EMPLOYEE NAME              EXPERTISE     SKILL
                                          LEVEL

   AA        KATHERINE O'HEARN             03        FORTRAN
   AA        KATHERINE O'HEARN             03        COBOL
   AA        KATHERINE O'HEARN             03        ASSEMBLER
   AA        KATHERINE O'HEARN             02        IDD
   BB        LAURA     PENMAN             (02)      (IDD)
   AA        LARRY     LITERATA            03        EDITING
   AA        LARRY     LITERATA            02        INTERACT
   AA        LARRY     LITERATA            02        COBOL
```

Example 3: Multiple Primary Paths

```
PATHAA EMPLOYEE EXPERTISE SKILL
PATHCC EMPLOYEE
```

In the above example, CA Culprit begins by navigating the path with the lowest path id, AA. Once CA Culprit retrieves all occurrences of path AA, CA Culprit navigates path CC before accessing the next occurrence of the EMPLOYEE record. The contents of the input buffer appear below:

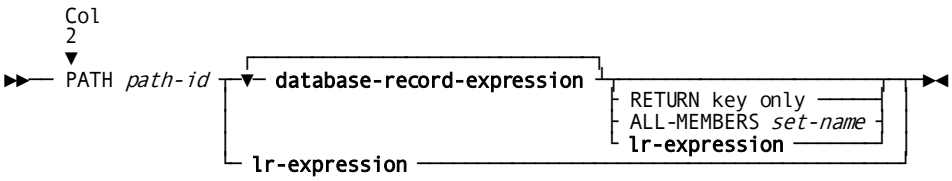
```
PATH ID      EMPLOYEE NAME              EXPERTISE     SKILL
                                          LEVEL

   AA        KATHERINE O'HEARN            03        FORTRAN
   AA        KATHERINE O'HEARN            03        COBOL
   AA        KATHERINE O'HEARN            03        ASSEMBLER
   AA        KATHERINE O'HEARN            02        IDD
   CC        KATHERINE O'HEARN           (02)      (IDD)
   CC        LAURA     PENMAN            (02)      (IDD)
   AA        LARRY     LITERATA          03         EDITING
   AA        LARRY     LITERATA          02        INTERACT
   AA        LARRY     LITERATA          02        COBOL
   CC        LARRY     LITERATA         (02)      (COBOL)
```

PATH Parameter Syntax

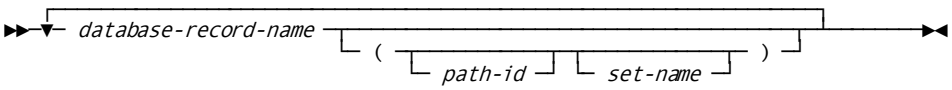
Purpose

Defines a path through the database.

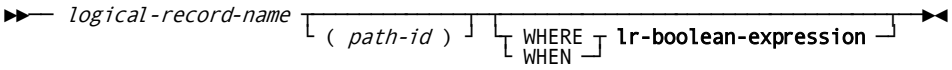
Syntax



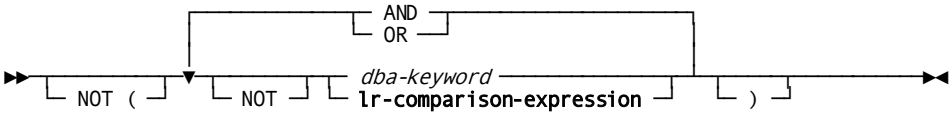
Expansion of Database-record-expression



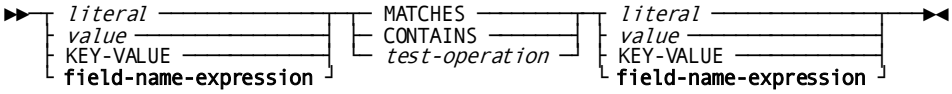
Expansion of lr-expression



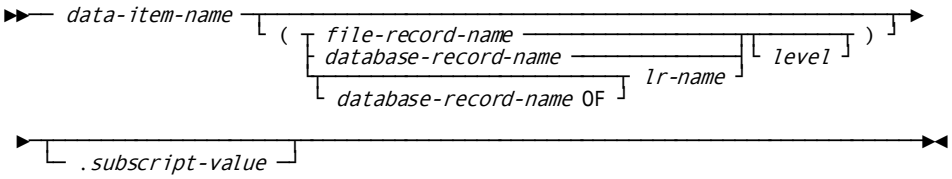
Expansion of lr-boolean-expression



Expansion of lr-comparison-expression



Expansion of Field-name-expression



Syntax Rules

PATH

Identifies the parameter type. It must be coded starting in column 2. More than one PATH parameter can be specified in a CA Culprit run.

path-id

A unique 2-character alphanumeric string that identifies the primary path. *Path-id* must be coded starting in column 6; the default is **. If the CA Culprit run accesses a table, CA Culprit automatically generates a PATH parameter that specifies 01 as the path id.

For considerations that apply to *path-id*, see Specifying Path Identifiers (see page 222).

database-record-expression

Specifies a database record and optionally specifies an alternate path ID and set relationship. *Database-record-expression* can be repeated as often as required to construct a path, provided that no more than 100 record types appear on a single path and no more than 255 record types are specified for all the PATH parameters defined for the run.

See expanded syntax for *database-record-expression* below.

RETURN KEY ONLY

A keyword expression that can be specified only if the entry record is accessed using an index. If specified, RETURN KEY ONLY must be the last entry on the path.

When a PATH parameter specifies RETURN KEY ONLY, CA Culprit does the following:

- Accesses only the entry record on the PATH parameter
- Retrieves the indexed field of the entry record; CA Culprit does not retrieve the entry record itself
- Returns the indexed field and the db-key of the entry record to the input buffer

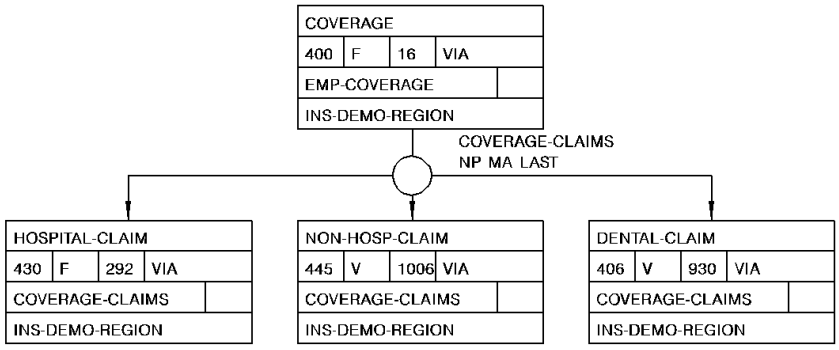
The RETURN KEY ONLY option can be used to create a key file that contains index-key values or to compare key file values with database values.

ALL-MEMBERS *set-name*

Applies to multiple-member set relationships only. Optionally, *set-name* may be enclosed in parentheses.

The following figure illustrates a multiple-member set relationship.

Example of a Multiple-Member Set



Note: The COVERAGE record is the owner of three types of member records: HOSPITAL-CLAIM, NON-HOSP-CLAIM, and DENTAL-CLAIM. COVERAGE-CLAIMS is the name of the set relationship.

ALL-MEMBERS instructs CA Culprit to access each occurrence of each record in a multiple-member set. *Set-name* specifies the name of the multiple-member set; the name must be enclosed in parentheses.

The following coding considerations apply to the ALL-MEMBERS option:

- ALL-MEMBERS must be specified after the name of the owner record in the multiple-member set.
- ALL-MEMBERS must be the last entry on the path.
- ALL-MEMBERS cannot be used when the path contains a logical record.

To access data from more than one but not all member record types, the PATH parameter must specify the ALL-MEMBERS option. The user's procedure logic or selection criteria must control processing based on the value of REC-NAME. REC-NAME indicates which member record type has been returned to the input buffer for a given string. For more information about REC-NAME, see Database Field Name References (see page 244).

Note: When the ALL-MEMBERS option is specified, you must allocate enough storage space in the input buffer for each member record in the multiple-member set. For more information about specifying the size of the input buffer, see INPUT Parameter (see page 43).

lr-expression

Indicates the name of a logical record following a string of database records on the PATH parameter. A logical record can follow database records on the PATH parameter under the following circumstances:

- The database records are defined to the logical record subschema.
- The usage mode defined for the logical record subschema is USAGE IS MIXED.
- The logical record retrieval path requires a particular set currency to be in effect before CA Culprit retrieves the logical record. In order to establish currency, the PATH parameter can specify a path with one or more database records preceding the logical record.
- A database record contains a value that is referenced in the WHERE clause of the logical record expression, as shown below:

```
 PATHAA EMPLOYEE EMP-JOB-LR
*       WHERE EMP-ID-0415 EQ EMP-ID-0415 (EMPLOYEE)
```

See expanded syntax for *lr-expression* later in this chapter.

Expansion of Database-record-expression

database-record-name

Identifies a name of a database record. At run time, CA Culprit accesses the entry record occurrences in one of the following ways:

- If KEYFILE or KEY parameters are specified, CA Culprit retrieves each occurrence in order as specified in the key file or on the KEY parameter, respectively.
- If the PATH parameter specifies an index set, CA Culprit retrieves each occurrence in index-key order.
- If neither of the above is specified, CA Culprit retrieves each occurrence as it is encountered in the database.

For rules that apply to the order of database records in a path definition, see Defining a Path.

path-id

Identifies either a null path or an alternate path. *Path-id* is an optional 2-character alphanumeric value that must be unique for the run and be enclosed in parentheses. If the PATH parameter also specifies *set-name* (see below), both *path-id* and *set-name* must be enclosed in the parentheses.

CA Culprit moves *path-id* to the PATH-ID field in the input buffer when one of the following conditions is true:

- CA Culprit can follow the path as far as the record type preceding that for which the path ID is specified. In the example below, CA Culprit returns BB to the PATH-ID field when it returns all records, excluding SKILL, to the input buffer:

  ```
  PATHAA DEPARTMENT EMPLOYEE EXPERTISE SKILL (BB)
  ```

  In the special case of an entry record, either the path contains no occurrences of the entry record or the database contains no occurrences of the entry record for a particular key.

- CA Culprit can follow the path as far as the record type for which the path ID is specified, but selection criteria eliminate the occurrence of the record. In the example below, CA Culprit returns BB to the PATH-ID field each time the employee ID does not equal 0301 or 0054:

  ```
  PATHAA DEPARTMENT EMPLOYEE (BB) EXPERTISE SKILL
  SELECT EMPLOYEE WHEN EMP-ID-0415 EQ (0301 0054)
  ```

set-name

An optional value that specifies the name of a set relationship between the previous record type and the current record type; the set must be defined to the subschema. If specified, *set-name* must be enclosed in parentheses. If the PATH parameter also specifies *path-id* (see above), both *path-id* and *set-name* must be enclosed in the parentheses.

The following considerations apply to *set-name*:

- In the special case of an entry record, *set-name* must be specified if the entry record is indexed and is to be accessed using the index; the name of the set must be the index set name.
- If more than one set relationship exists between the previous record type and the current record type, a set name must be specified.

Expansion of lr-expression

logical-record-name

The name of a logical record defined in the subschema named on the INPUT parameter. *Logical-record-name* must be the only record on the path.

The following considerations apply to logical record retrieval:

- Each automatic retrieval path must specify the same logical record; only one can specify a WHERE clause. The WHERE clause is described in greater detail later in this chapter.
- Each dummy path can specify a different logical record and a WHERE clause.

path-id

An optional 2-character alphanumeric value unique to the run; if specified, it must be enclosed in parentheses. CA Culprit moves the value of *path-id* to the PATH-ID field in the input buffer when one of the following conditions is true:

- No logical record occurrences meet the criteria specified in the WHERE clause. CA Culprit returns LR-NOT-FOUND or LR-ERROR to the LR-STATUS field in the input buffer the first time an attempt is made to retrieve the logical record.
- A KEY or KEYFILE parameter specifies a key value and no logical record meets the criteria specified in the WHERE clause for the key value. If a KEY parameter specifies the key value, the value will be in the appropriate field in the logical record in the input buffer unless the DBA-specified retrieval path overlays the field. If a KEYFILE parameter specifies the key value, the key file record will be in the input buffer.
- A logical record occurrence does not meet the selection criteria specified on a SELECT/BYPASS parameter for that record. The input buffer will contain the logical record.

Note: If the DBA-specified logical record status is other than LR-NOT-FOUND or LR-ERROR, CA Culprit continues as if a status of LR-FOUND were returned. The user can test the reserved word LR-STATUS to determine the logical record status that was returned.

WHERE/WHEN

A keyword that introduces selection criteria for logical record retrieval.

lr-boolean-expression

Specifies criteria to be applied toward the selection of logical record occurrences. WHEN may be used as a synonym for WHERE.

See expanded syntax for *lr-boolean-expression* later in this chapter.

Expansion of lr-boolean-expression

NOT

A keyword that reverses the logic of the boolean expression; for example, WHERE NOT DEPT-ID EQ '3200' returns all logical record occurrences that do not specify 3200 for DEPT-ID.

dba-keyword

A keyword that is used in the logical record definition and defined to the data dictionary.

AND/OR

Logical connectors that allow the user to state complex logical record selection criteria. With AND, both conditions must be true; with OR, one condition must be true.

lr-comparison-expression

A boolean expression, which is expanded below.

Expansion of lr-comparison-expression

literal

A value to be compared. It is an alphanumeric, numeric, or hexadecimal literal enclosed in single quotation marks, as follows:

Type of literal	How to code it	Example
alphanumeric	A 1- to 64-character alphanumeric value that consists of letters, numbers, and special characters in any combination. To express an apostrophe (a single quotation mark), code two consecutive single quotation marks.	'ERNIE''S DINER'
numeric literal	A 1- to 15-character numeric value, optionally preceded by a sign and optionally containing an embedded or trailing decimal point.	'12.34'
hexadecimal	A 1- to 50-character (25-byte) hexadecimal value, preceded by X.	X'010203'

value

Specifies an arithmetic operation that contains operands and operators.

An **operand** must be a numeric literal or the name of a numeric field. The left operand is separated from the right operand by an arithmetic operator (see below). A space must exist between each operand and operator.

An **operator** indicates the type of arithmetic calculation, as follows:

Operator	What it means
+	The value of the left operand is added to the value of the right operand.

Operator	What it means
-	The value of the right operand is subtracted from the value of the left operand.
*	The value of the left operand is multiplied by the value of the right operand.
/	The value of the left operand is divided by the value of the right operand.

Parentheses can be used within the arithmetic expression; if parentheses are not used, CA Culprit follows algebraic rules of precedence.

KEY-VALUE

Specifies that a value in a KEY or KEYFILE parameter is to be compared to a field in the logical record.

field-name-expression

Specifies a field in either the logical record or in the input buffer. Unless the field is qualified by a record name or a level number that establishes another meaning, the expression applies to a field in the logical record.

The expression applies to a field in the input buffer if CA Culprit retrieves a logical record on another path, a database record on this or another path, or a key file record.

In all cases, the field name must be defined to the data dictionary; the name cannot be defined on either a user-supplied REC parameter or a work field parameter.

See expanded syntax for *field-name-expression* below:

MATCHES

This is a comparison operator that performs either generic searches or pattern matching. The values of the left and right operands can be either literals or fields containing character strings or zoned decimal values. Literals, including numeric values, must be enclosed in single quotation marks.

When CA Culprit encounters MATCHES, it compares the left operand with the right operand, one character at a time, starting with the leftmost character. If a character in the left operand does not match its respective character in the right operand, the result of the comparison expression is false. If a match occurs for each character in the shorter operand, the result is true.

A **generic search** occurs when the right operand is shorter than the left. For example, WHERE CUST-NAME MATCHES 'B' retrieves each logical record occurrence for which the customer's name begins with the letter B.

Pattern matching occurs when the right operand includes special characters, as follows:

Special Character	Matches...
@	Any alphabetic character.
#	Any numeric character.
*	Any character.

For example, WHERE TELEPHONE-NUM MATCHES '617###4567' retrieves each logical record that contains a telephone number that begins with 617, ends with 4567, and contains any three digits in between.

CONTAINS

This comparison operator searches for an occurrence of the right operand in the left operand. The length of the left operand must be equal to or greater than the length of the right operand.

The values of the left and right operands can be either literals or fields containing character strings or zoned decimal values. Literals, including numeric values, must be enclosed in single quotation marks.

If the left operand does contain the substring, the result of the comparison expression is true; otherwise, the result is false. For example, WHERE CODE-NUM CONTAINS 'X23D' retrieves all logical record occurrences that have a code number containing X23D.

test-operation

Specifies operators to compare the value of the left operand to the value of the right operand, as follows:

Test Operation	Meaning	Synonyms
EQ	equal	E or =
NE	not equal	N or #
GT	greater than	H or >
LT	less than	L or <
GE	greater than or equal to	>= or =>
LE	less than or equal to	<= or =<

Expansion of Field-name-expression

data-item-name

The name of a field defined to the data dictionary.

file-record-name

Specifies the name of a key file record.

File-record-name can be specified only if the CA Culprit code includes a KEYFILE parameter that specifies the FN= keyword expression. This keyword expression indicates that the key file is defined to the data dictionary.

database-record-name

Specifies the name of a database record that precedes the logical record on the PATH parameter. The name of the database record is required only if the field name occurs in more than one type of record on the PATH parameter(s).

lr-name

Specifies the name of the logical record. The name is required only if the field name occurs in more than one type of record on the PATH parameter(s).

If the field name occurs in more than one type of record within the logical record, *lr-name* must be qualified as *database-record-name* OF *lr-name*, where *database-record-name* refers to a database record within the logical record.

level

Identifies a specific occurrence of a record in the input buffer, as described in REC Parameter (see page 212). If specified, *level* must be separated from the name of the record by a comma or a space and must precede a closing parenthesis.

Level has a special meaning when it qualifies a field in a WHERE clause associated with a logical record on a dummy path, as follows:

- If *level* does not qualify the field name in the WHERE clause, the field name references the value in the logical record. In the example shown below, EMP-ID-0415 refers to a value in EMP-JOB-LR because the field is not qualified:

  ```
  PATH-- EMP-JOB-LR WHERE EMP-ID-0415 EQ '0073'
  ```

- If *level* qualifies the field name in the WHERE clause, the field name references a value already in the input buffer when CA Culprit retrieves the logical record. Generally, this applies if CA Culprit retrieves the logical record by using the DB-EXIT facility; a DB-EXIT SET-VALUE call sets the value in the input buffer. In the example below, CA Culprit compares the value of EMP-ID-0415 in the logical record to a value that is already in the input buffer:

  ```
  PATH-- EMP-JOB-LR WHERE EMP-ID-0415 EQ EMP-ID-0415 (1)
  ```

subscript-value

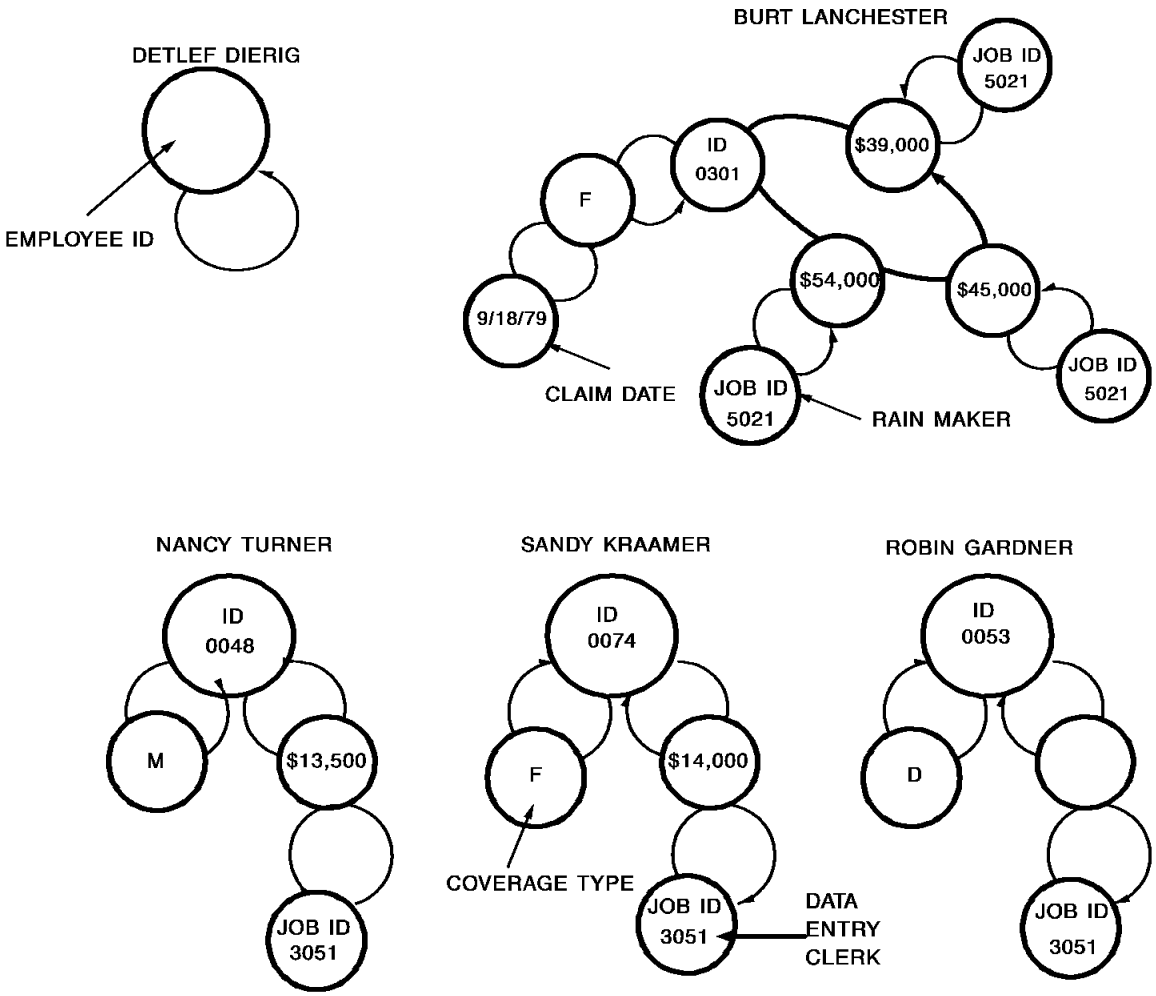
Identifies a specific occurrence of a multiply-occurring field; it is a value in the range 1 through 32,767. If specified, it must be separated from the name of the multiply-occurring field by a period (.).

Examples

Sample PATH parameters are shown and described in following figure.

Employee Database Subschema (see page 415) contains a figure of the database structure and lists the database field definitions. Examples 1 through 4 refer to the following set occurrence diagram.

The logical record examples (examples 9 through 11) access a logical record that contains certain fields from the DEPARTMENT, OFFICE, EMPLOYEE, and JOB records.



Note: The set occurrence diagram contains the following types of information: the employee's name (EMP-NAME-0415) and ID (EMP-ID-0415) from the EMPLOYEE record, the employee's salary (SALARY-AMOUNT-0420) from the EMPOSITION record, the employee's job title (TITLE-0440) and ID (JOB-ID-0440) from the JOB record, the employee's insurance coverage plan (TYPE-0400) from the COVERAGE record, and any hospital claims (CLAIM-DATE-0430) from the HOSPITAL-CLAIM record.

Example 1: Primary Path ID

```
PATHAA EMPLOYEE EMPOSITION JOB
```

CA Culprit accesses the first EMPLOYEE record occurrence and then searches for an associated EMPOSITION record occurrence followed by a JOB record occurrence. Once it completes a string and returns it to the input buffer, CA Culprit looks for the next JOB record occurrence associated with the EMPOSITION record. If it finds no more JOB records, CA Culprit looks for the next EMPOSITION record occurrence associated with the EMPLOYEE record, and so on. Since the PATH parameter specifies a primary path ID only, each string CA Culprit returns to the input buffer must contain an occurrence of the EMPLOYEE, EMPOSITION, and JOB records.

CA Culprit returns the following strings to the input buffer, based on the set occurrences

```
PATH-ID      EMPLOYEE            EMPOSITION      JOB

  AA     0301 BURT LANCHESTER     $54,500      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $45,000      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $39,000      5021 RAINMAKER
  AA     0048 NANCY TERNER        $13,500      3051 DATA ENTRY CLERK
  AA     0074 SANDY KRAAMER       $14,000      3051 DATA ENTRY CLERK
  AA     0053 ROBIN GARDNER       $14,000      3051 DATA ENTRY CLERK
```

Example 2: Null Path Id

```
KEY EMP-ID (0312 0301 0048 0074 0053 0444)
PATHAA EMPLOYEE (BB) EMPOSITION JOB
```

The KEY parameter specifies employee ID values with which CA Culprit can access each EMPLOYEE record in path AA directly. Following the EMPLOYEE record on the PATH parameter is a null path ID (BB). If CA Culprit does not find an occurrence of an EMPLOYEE record by using the ID values specified in the KEY parameter, it returns BB to the PATH-ID field in the input buffer.

The following sequence of strings returned to the input buffer indicates that there is no employee with an ID of 0312. Also note that CA Culprit did not return a string to the input buffer for employee 0444, which indicates that CA Culprit was not able to complete path AA for this individual.

```
PATH-ID      EMPLOYEE            EMPOSITION      JOB

  BB     0312
  AA     0301 BURT LANCHESTER     $54,500      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $45,000      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $39,000      5021 RAINMAKER
  AA     0048 NANCY TERNER        $13,500      3051 DATA ENTRY CLERK
  AA     0074 SANDY KRAAMER       $14,000      3051 DATA ENTRY CLERK
  AA     0053 ROBIN GARDNER       $14,000      3051 DATA ENTRY CLERK
```

Example 3: Alternate Path Id

```
KEY EMP-ID (0312 0301 0048 0074 0053 0444)
PATHAA EMPLOYEE (BB) EMPOSITION (CC) JOB
```

This PATH parameter specifies three path identifiers. Path AA is the primary path id; path BB is the null path id; and path CC is an alternate path ID for paths that contain an EMPLOYEE record occurrence but no occurrence of the EMPOSITION record.

As shown in the following sequence of strings returned by CA Culprit to the input buffer, CA Culprit returns CC to the PATH-ID field for the DETLEF DIERIG record occurrence. The remainder of the input buffer contains residual salary and job information associated with the ROBIN GARDNER record occurrence. The user's procedure logic must test the value of PATH-ID to determine which path CA Culprit navigated.

```
PATH-ID      EMPLOYEE            EMPOSITION      JOB

  BB     0312
  AA     0301 BURT LANCHESTER     $54,500      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $45,000      5021 RAINMAKER
  AA     0301 BURT LANCHESTER     $39,000      5021 RAINMAKER
  AA     0048 NANCY TERNER        $13,500      3051 DATA ENTRY CLERK
  AA     0074 SANDY KRAAMER       $14,000      3051 DATA ENTRY CLERK
  AA     0053 ROBIN GARDNER       $14,000      3051 DATA ENTRY CLERK
  CC     0444 DETLEF DIERIG      ($14,000)    (3051 DATA ENTRY CLERK)
```

Example 4: Multiple vs. Single Paths

```
PATHAA EMPLOYEE EMPOSITION
PATHDD EMPLOYEE COVERAGE HOSPITAL-CLAIM
 or
PATHAA EMPLOYEE EMPOSITION EMPLOYEE COVERAGE HOSPITAL-CLAIM
```

These PATH parameters are alternative ways to access two record types that are related to the same owner by different set relationships, as follows:

- The two-path method returns each EMPOSITION record occurrence and each set of COVERAGE/HOSPITAL-CLAIM record occurrences once per employee. CA Culprit returns the following strings to the input buffer for the two-path method:

```
PATH-ID      EMPLOYEE          EMPOSITION  COVERAGE  HOSPITAL-CLAIM

  AA     0301 BURT LANCHESTER   $54,500       -          -
  AA     0301 BURT LANCHESTER   $45,000       -          -
  AA     0301 BURT LANCHESTER   $39,000       -          -
  DD     0301 BURT LANCHESTER  ($39,000)      F       09/18/79
  AA     0048 NANCY TERNER      $13,500      (F)     (09/18/79)
  AA     0074 SANDY KRAAMER     $14,000      (F)     (09/18/79)
  AA     0053 ROBIN GARDNER     $14,000      (F)     (09/18/79)
```

- The one-path method retrieves each set of COVERAGE/HOSPITAL-CLAIM record occurrences once per EMPOSITION record occurrence. This method is less efficient than the two-path method. It can also affect the strings returned by CA Culprit to the input buffer, as follows:
  - Creates duplicate COVERAGE/HOSPITAL-CLAIM entries on reports when more than one EMPOSITION occurrence exists for an employee, as in the case of the BURT LANCHESTER record occurrence. The user's procedure logic can be designed to control duplicate entries during string generation.
  - Eliminates strings that would otherwise be processed in the two-path method, unless the PATH parameter includes alternate ids.

  CA Culprit returns the following strings to the input buffer for the one-path method:

```
PATH-ID      EMPLOYEE          EMPOSITION  COVERAGE  HOSPITAL-CLAIM

 AA     0301 BURT LANCHESTER   $54,500       F      09/18/79
 AA     0301 BURT LANCHESTER   $45,000       F      09/18/79
 AA     0301 BURT LANCHESTER   $39,000       F      09/18/79
```

Example 5: RETURN KEY ONLY Path

```
PATH01 JOB (JOB-TITLE-NDX) RETURN KEY ONLY
0151*010 TITLE-0440
0151*020 DBKEY-PAGE   F0
0151*030 DBKEY-LINE   F0
```

CA Culprit accesses each JOB record occurrence by using the index defined for that record; the record is indexed by job titles. RETURN KEY ONLY instructs CA Culprit to return the index key and the db-key of each JOB record occurrence to the input buffer. Following is a portion of the output for Report 01:

```
   TITLE                 DBKEY-PAGE      DBKEY-LINE

 ACCOUNTANT               75113             2
 AP-CLERK                 75114             5
 AR-CLERK                 75149             5
 COMPUTER OPERATOR        75147             3
    -                       -               -
    -                       -               -
```

Example 6: ALL-MEMBERS Path

```
PATH03 COVERAGE ALL-MEMBERS (COVERAGE-CLAIMS)
```

The code specified in path 03 instructs CA Culprit to navigate the COVERAGE-CLAIMS multiple-member set relationship that appears in the figure under **Example of a Multiple-Member Set**. For each COVERAGE record occurrence, CA Culprit accesses either a HOSPITAL-CLAIM, a NON-HOSP-CLAIM, or a DENTAL-CLAIM record occurrence. CA Culprit fills the REC-NAME field in the input buffer with the name of the member record type retrieved. The strings returned by CA Culprit appear below:

```
    REC-NAME       COVERAGE    DENTAL       HOSPITAL     NON-HOSPITAL

NON-HOSP-CLAIM        F          -             -           10/07/82
NON-HOSP-CLAIM        F          -             -           08/18/81
HOSPITAL-CLAIM        M          -          11/07/82     (08/18/81)
DENTAL-CLAIM          F       05/23/77      (11/07/82)   (08/18/81)
DENTAL-CLAIM          F       04/10/80      (11/07/82)   (08/18/81)
HOSPITAL-CLAIM        F      (04/10/80)     09/18/79     (08/18/81)
```

Example 7: ALL-MEMBERS Path and REC-NAME

```
PATH03 COVERAGE ALL-MEMBERS (COVERAGE-CLAIMS)
SELECT BUFFER WHEN REC-NAME EQ ('NON-HOSP-CLAIM' 'DENTAL-CLAIM')
```

At run time, CA Culprit accesses all three COVERAGE-CLAIMS record types, but the SELECT parameter limits record retrieval to only the NON-HOSP-CLAIM and DENTAL-CLAIM record occurrences. The strings that CA Culprit returns to the input buffer appear below:

```
   REC-NAME       COVERAGE    DENTAL       HOSPITAL    NON-HOSPITAL

NON-HOSP-CLAIM        F          -             -           10/07/82
NON-HOSP-CLAIM        F          -             -           08/18/81
DENTAL-CLAIM          F       05/23/77      (11/07/82)   (08/18/81)
DENTAL-CLAIM          F       04/10/80      (11/07/82)   (08/18/81)
```

Example 8: Bill-of-Materials Path

```
PATH32 DEPARTMENT EMPLOYEE STRUCTURE (REPORTS-TO) EMPLOYEE (MANAGES)

0151*010 DEPT-ID      HH 'DEPT ID'
0151*020 EMP-NAME     HH 'EMPLOYEE'
0151*030 EMP-NAME (2) HH 'MANAGER'
```

Once CA Culprit accesses a DEPARTMENT record occurrence, it navigates a bill-of-materials data structure. In this example, CA Culprit retrieves each employee in the department and the people that manage the employee. Since the EMPLOYEE record appears twice on the PATH parameter, all field names that pertain to the second occurrence of the record in the input buffer must be qualified by a level indicator. A portion of the report output appears below:

```
DEPT ID     EMPLOYEE              MANAGER

 0100       JOHN  ROTGUT          HENRIETTA HENDON
 0100       PAM   VAN DONG        ROBBY WILDER
 0100       PAM   VAN DONG        HENRIETTA HENDON
 0100       ROBBY WILDER          HENRIETTA HENDON
  -              -                     -
```

If REPORTS-TO and MANAGES were reversed in this example, CA Culprit would retrieve each employee in the department and the people that report to the employee, as shown in the following example:

```
DEPT ID     EMPLOYEE              STAFF

 0100       JOHN  ROTGUT          ELEANOR PEOPLES
 0100       JOHN  ROTGUT          HERBIE BABIE
 0100       PAM   VAN DONG        ROGER WILCO
 0100       PAM   VAN DONG        DANIEL MOON
 0100       PAM   VAN DONG        RENE MAKER
 0100       ROBBY WILDER          PAM VAN DONG
   -              -                     -
```

Example 9: Logical Record Path Using KEY-VALUE

```
 PATHA1 EMP-JOB-LR WHERE JOB-ID-0440 EQ '3051'
*                   AND EMP-ID-0415 EQ KEY-VALUE
 KEY EMP-ID-0415 (0312 0301 0048 0074 0053 0444)
```

This PATH parameter specifies retrieval of EMP-JOB-LR logical record occurrences. The WHERE clause instructs CA Culprit to retrieve those logical records that have a job ID of 3051 and employee ids that equal the values specified on the KEY parameter. CA Culprit returns the following logical record strings to the input buffer:

```
  EMP-ID     EMP-NAME       JOB-ID      TITLE

  0048     NANCY TERNER     3051     DATA ENTRY CLERK
  0053     ROBIN GARDNER    3051     DATA ENTRY CLERK
  0074     SANDY KRAAMER    3051     DATA ENTRY CLERK
```

Example 10: Logical Record Path Using CONTAINS

```
 PATHA2 EMP-JOB-LR (JOB-TITLE-NDX)
*                  WHERE TITLE-0440 CONTAINS 'ENTRY'
*                  AND (EMP-ID-0415 EQ '0048' OR
*                       EMP-ID-0415 EQ '0053' OR
*                       EMP-ID-0415 EQ '0074')
```

CA Culprit accesses EMP-JOB-LR logical record occurrences by using the index defined for the JOB record type. The WHERE clause selects those logical record occurrences where the job title contains the word ENTRY and the employee ID equals the listed values. CA Culprit returns the same strings to the input buffer as in the previous example.

Example 11: Logical Record Using MATCHES

```
PATHAA EMP-JOB-LR WHERE EMP-ID-0415 MATCHES '03##'
```

CA Culprit returns only those logical record occurrences where the employee ID value begins with 03 and ends with any other combination of numbers, as shown below:

```
0349   ROGER     WILCO
0371   BETH  M.  CLOUD
0334   CAROLYN   CROW
0321   DANIEL    MOON
0301   BURT      LANCHESTER
0366   ALAN      DONOVAN
0355   MARK      TIME
```

More information:

KEY and KEYFILE Parameters (see page 251)
INPUT Parameter (see page 203)
The DB-EXIT Facility (see page 429)

Database Field Name References

Uses

Report-specific, SELECT/BYPASS, and PATH parameters can reference the following types of field names:

- Reserved keywords that define fields in the CA Culprit input buffer
- Fields defined to the data dictionary or on a user-supplied REC parameter, which must be qualified by record type or level number

Each type of field name reference is discussed separately below.

Reserved Keywords

Reserved keywords either apply to runs that access database and logical records or runs that access tables. Each category is discussed separately below.

Accessing Database and Logical Records

The first 38 bytes of the CA Culprit input buffer contain fields that can be used to determine the contents of the input buffer; the input buffer also contains the db-key value for each record in the input buffer. Users can access these fields by using reserved keywords.

Reserved keywords can be referenced on SORT, edit, process, and SELECT/BYPASS parameters. num=21.Database-Specific Process Parameters, identifies the appropriate keywords for each type of process operation.

Keywords

Each keyword is described separately below:

PATH-ID

Contains the 2-byte alphanumeric path ID of the records currently in the input buffer. The user's procedure logic can test the value of PATH-ID to determine which string CA Culprit returned to the input buffer.

REC-NAME

Contains the 16-byte name of the member record currently being processed; this reserved word applies only when CA Culprit accesses records that participate in a multiple-member set relationship. The user's procedure logic can test the value of REC-NAME to determine which member record type is in the input buffer. The value of REC-NAME is the record name defined to the subschema; it cannot be the record synonym name.

IDMS-STATUS

Contains a 4-byte alphanumeric CA IDMS/DB status code; this value pertains to DB-EXIT retrievals only.

LR-STATUS

Contains a 16-byte alphanumeric status of the most recently retrieved logical record. Possible values for LR-STATUS are the following:

- LR-FOUND
- LR-NOT-FOUND
- LR-ERROR

If the logical record status is anything other than the three values listed above, CA Culprit continues as if a status of LR-FOUND were returned.

DBKEY

References a 4-byte binary field containing the database key for the most recently retrieved occurrence of each record type specified on a PATH parameter. This field immediately follows the corresponding record in the input buffer.

DBKEY-PAGE

References the page number of the db-key, as defined in the schema syntax. CA Culprit generates a multi-bit binary field (data type 5).

Note: For more information on database key structures, see the *CA IDMS Database Administration Guide*.

DBKEY-LINE

References the line number of the db-key, as defined in the schema syntax. CA Culprit generates a multi-bit binary field (data type 5).

Note: For more information on database key structures, see the *CA IDMS Database Administration Guide*.

DBKEY-ALPHA

References a 4-byte alphanumeric field that redefines DBKEY. This keyword is useful for writing db-key values to a file.

DBKEY, DBKEY-PAGE, and DBKEY-LINE should be treated as zoned decimal values rather than as binary values; CA Culprit converts the values automatically. Any of the db-key reserved words can be qualified by record name and level, as described later in this chapter under Qualified Field Name References (see page 247). An unqualified reference defaults to the entry record on the PATH parameter.

More information:

The DB-EXIT Facility (see page 429)

Accessing Tables

In a CA Culprit run that consolidates one or more tables, the first 95 bytes of the CA Culprit input buffer contain fields that can be used to determine the table associated with the row that is in the input buffer. Users can access these fields by using reserved keywords.

Reserved keywords can be referenced on SORT, edit, process, and SELECT/BYPASS parameters. Database-Specific Process Parameters **(see page 283)** identifies the appropriate keywords for each type of process operation.

Keywords

Each keyword is described separately below:

TABLE-ID

Is a 3-byte zoned decimal field that indicates which data table is currently being accessed. Ids are assigned in the order the tables are read. The primary table ID is 001, the second table ID is 002, and so on.

TABLE-NAME

Is a 56-byte alphanumeric field that contains the table name supplied on the INPUT parameter TABLE= keyword.

SUBSCHEMA-NAME

Is an 8-byte alphanumeric field that contains the subschema name of the table currently being read.

Qualified Field Name References

A field name must be qualified if the field name is **not** unique. A field name is not unique under the following circumstances:

- If it occurs in more than one of the following record types:
  - Database records named on the PATH parameter
  - Logical records named on the PATH parameter
  - Records of a file characterized on the KEYFILE parameter
  - Records of a file characterized on an INPUT parameter
- If it is defined for two or more columns in a view of a table.

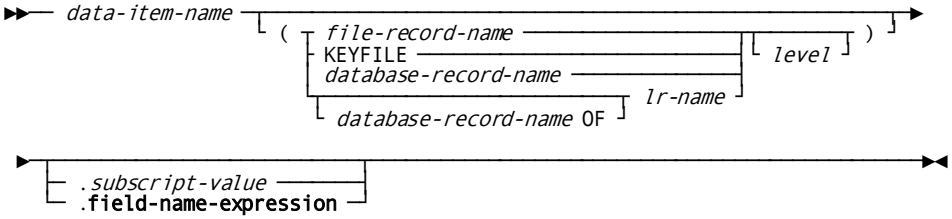
The field name qualifiers are considered part of the field name.

Field-name-expression

Purpose

Uniquely identifies a field name.

Syntax



Syntax Rules

Syntax rules follow:

data-item-name

Is the name of a field in a file defined to the data dictionary, a file characterized on the KEYFILE parameter, a database record, a logical record, or a table.

file-record-name

Is the name of a record defined to the data dictionary. The record must belong to the file named on the FN= keyword expression of an INPUT or KEYFILE parameter.

KEYFILE

Specifies that the field is associated with the key file defined on the KEYFILE parameter.

database-record-name

Specifies the name of a database record that appears on a PATH parameter.

In a run that accesses a view of a table, *database-record-name* specifies the name of the record that defines a source table in the view definition. *Database-record-name* is always RFUR-*nnnnnn*-DATA, where *nnnnnn* is the table definition number (TDN) generated when the table was defined.

To determine the table definition number, check either of the following:

- ASF's Table Definition screen (DEFN NUMBER field); pad the value with zeros (0) on the left, if necessary.
- CA Culprit's INPUT parameter report on the Input Parameter Listing. The TDN is the six digits that form part of the subschema name.

logical-record-name

Specifies the name of the logical record on the PATH parameter.

If the field is defined to two or more records within one logical record, the field must be qualified by **database-record-name OF logical-record-name**. For example, EMP-ID (EMPLOYEE OF EMP-JOB-LR) references the EMP-ID field in the EMPLOYEE record of the logical record.

In a run that accesses a view of a table, *logical-record-name* is the logical record name of the table view. To determine the logical record name, check either:

- ASF's Extended Table Definition screen (LOGICAL RECORD NAME field)
- CA Culprit's INPUT parameter report on the Input Parameter Listing (LR-NAME field)

If one of these qualifiers is specified, it must be enclosed in parentheses.

level

Specifies an occurrence of the field in the input buffer. *Level* is a value in the range 1 through 256; the default is 1. If specified, *level* must be separated from the name of the record or file (if any) by a comma or a space and must precede a closing parenthesis. *Level* is determined as follows:

- When *data-item-name* refers to a database or logical record, the level is established as follows:
  - If the record appears more than once on a single PATH parameter, the first occurrence of the record is level 1, the second is level 2, and so on.
  - If the record appears on one or more dummy paths, the first occurrence of the record is one higher than the highest level for that record on an automatic retrieval PATH parameter. Each subsequent appearance on the same or another dummy path is one level higher than the previous appearance.

- When *data-item-name* refers to a field in a nondatabase record, *level* is required only during a match file run. If two or more INPUT or KEYFILE parameters specify the same file name on the FN= keyword expression, or if the record appears in two or more files named on the FN= keyword expression, the file named on the first FN= keyword expression is at level 1, the file named on the second expression is at level 2, and so on.

subscript-value

Specifies a numeric literal that identifies a specific occurrence of a multiply-occurring field; for example, EMP-NAME.4 identifies the fourth occurrence of EMP-NAME.

If a subscript value is provided, it must be separated from the name of the multiply-occurring field by a period (.). The value of the subscript should be an integer in the range 1 through *n*, where *n* is the number of repetitions of the multiply-occurring field.

For example if EMP-NAME occurs ten times, the value of the subscript should not exceed 10. The value of the subscript can be tested in procedure logic.

field-name-expression

Specifies the name of a singly-occurring numeric input or work field whose value identifies a specific occurrence of a multiply-occurring field; for example, if INDX has a value of 2, then EMP-NAME.INDX identifies the second occurrence of EMP-NAME. The subscript field definition must not specify a decimal point.

For more information on **field-name-expression**, see User-Defined Variables (see page 104).

Examples

The following examples of CA Culprit parameters illustrate the use of both reserved keywords and record and level qualifiers.

Example 1: Using PATH-ID in Procedure Logic

```
PATHAA EMPLOYEE (BB) EXPERTISE
017    IF PATH-ID EQ 'BB' 500
 -
017500 MOVE 'NO EMPLOYEE' TO MESSAGE
```

If CA Culprit does not find an occurrence of an employee, it returns BB to the PATH-ID field. The user's procedure logic tests the value of PATH-ID. If PATH-ID equals BB, CA Culprit moves a message to a work field for output.

Example 2: Using REC-NAME on an Edit Parameter

```
PATHF3 COVERAGE ALL-MEMBERS (COVERAGE-CLAIMS)
0151*010 REC-NAME
```

This PATH parameter instructs CA Culprit to navigate the COVERAGE-CLAIMS set illustrated in the 'Example of a Multiple-Member Set' schema available in the PATH Parameter (see page 227) section. When CA Culprit navigates the multiple-member set, it returns either HOSPITAL-CLAIM, NON-HOSP-CLAIM, or DENTAL-CLAIM to the REC-NAME field. The name of the member record can be output by coding REC-NAME on a type 5 edit parameter.

Example 3: Using TABLE-ID as a Subscript

```
INPUT TABLE=ACCOUNTING TYPE=COPY
INPUT TABLE=PERSONNEL TYPE=CONSOL
010 DEPARTMENT.2 'ACCOUNTING DEPARTMENT' 'PERSONNEL DEPARTMENT'
0151*010 DEPARTMENT.TABLE-ID
0151*020 EMP-NAME-0415
```

This example consolidates the ACCOUNTING and PERSONNEL tables. For each row of the ACCOUNTING table, TABLE-ID has a value of 1; for each row of the PERSONNEL table, TABLE-ID has a value of 2. CA Culprit uses these values to reference each occurrence of work field DEPARTMENT.

Example 4: Field Name Level Qualifiers

```
 PATHAA EMPLOYEE STRUCTURE (REPORTS-TO) EMPLOYEE (MANAGES)
 SELECT EMPLOYEE 2 IN PATH AA
*       WHEN EMP-NAME-0415(EMPLOYEE,2) EQ 'HERBIE     BABIE'
 0151*010 EMP-NAME-0415
 0151*020 EMP-NAME-0415 (EMPLOYEE,2)
```

Path AA contains two occurrences of the EMPLOYEE record; level 1 is the entry record and level 2 is the record retrieved using the MANAGES set relationship. The SELECT parameter tests the level 2 occurrence of the EMPLOYEE record for employee Herbie Babie. In the output shown below, column 1 contains the names of employees who report to Herbie Babie:

```
NANCY     TERNER                HERBIE    BABIE
ROBIN     GARDNER               HERBIE    BABIE
JANE      FERNDALE              HERBIE    BABIE
HERBERT   LIPSICH               HERBIE    BABIE
SANDY     KRAAMER               HERBIE    BABIE
```

Example 5: Field Name Level Qualifiers

```
INPUT 100 FN=ACCT-FILE-1
INPUT 100 FN=ACCT-FILE-2
0151*010  ACCT-NUMBER
0151*020  ACCT-STATUS(2)
```

This match-file run specifies two INPUT parameters. Each names an input file defined to the data dictionary. Because ACCT-FILE-1 and ACCT-FILE-2 have the same record structure, the input field names must be qualified by level number. The type 5 edit parameter that specifies ACCT-NUMBER refers to a field in a record within ACCT-FILE-1, the first file defined for the CA Culprit run. ACCT-STATUS(2) refers to a field within the second file, ACCT-FILE-2.

Example 6: Column Qualifiers in a Table View

```
INPUT TABLE=BUDGET
0151*020 EMP-NAME
0151*030 EMP-NAME(RFUR-000115-DATA OF BUDGET)
```

Table BUDGET is a view of two stored tables. The view has two columns with the same name, EMP-NAME. The first type 5 parameter references the first occurrence of EMP-NAME in the table view.

The second type 5 parameter qualifies the second occurrence of EMP-NAME in the table view. The values in parentheses refer to the database record name of the stored table in which the column occurs (RFUR-000115-DATA) and the name of the logical record that defines the view (BUDGET).

More information:

User-Defined Variables (see page 104)

KEY and KEYFILE Parameters

Purpose

KEY and KEYFILE parameters are optional parameters that identify unique **key values** CA Culprit uses to access the first record (that is, the entry record) on the PATH parameter. CA Culprit accesses only those record occurrences that contain the specified key values rather than each record as it is encountered in the database.

The following types of key values can be specified on KEY and KEYFILE parameters:

- Database keys
- Index keys
- CALC keys
- Logical record keys
- Table keys

KEY and KEYFILE Differences

KEY and KEYFILE parameters differ in how key values are specified. The KEY parameter specifies actual key values; the KEYFILE parameter identifies characteristics of a sequential file that contains key values.

Both parameters can be specified in the same CA Culprit run and perform the following functions:

- Select database entry record occurrences with specified key values. CA Culprit accesses the database record occurrences directly by the key value.
- Establish key values that can be used to select logical records or rows of a table.

  KEY-VALUE, the keyword that can be referenced in the WHERE clause on the PATH or INPUT parameter, automatically assumes the value established by the KEY or KEYFILE parameter. CA Culprit compares the key value to the criteria specified in the WHERE clause. If the key value meets the criteria, CA Culprit processes the logical record occurrence or table row. When no more logical record occurrences or rows meet the criteria, CA Culprit reads the next key value.

KEY and KEYFILE parameters are discussed separately in this chapter.

About the KEY Parameter

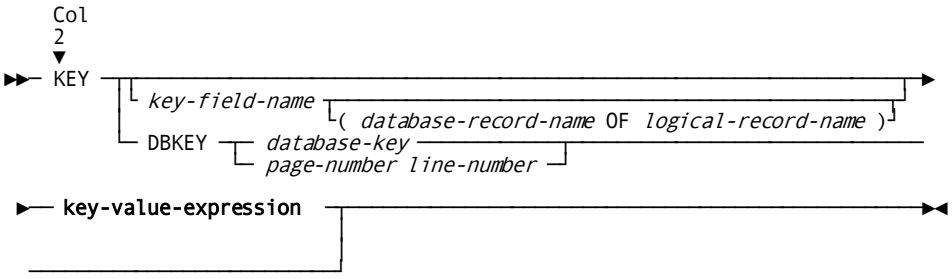
Refer to the syntax on the following page.

KEY Parameter

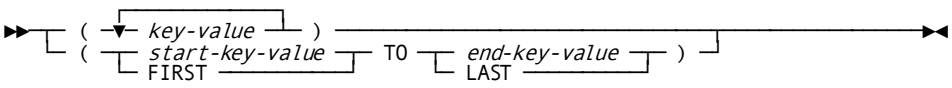
Purpose

The KEY parameter identifies either a field or column name and associated key values.

Syntax



Expansion of Key-value-expression



Syntax Rules

KEY

Specifies the type of parameter; it must be coded starting in column 2.

key-field-name

Specifies the name of the key field for the entry record. It is either a CALC-key field, index-key field, logical record key field, or a table column, as follows:

- The field is a CALC-key field if the entry record has a CALC location mode and is not accessed by using an index.
- The field is an index-key field if the entry record is accessed by using an index.
- The field is a logical record key field if the entry record is a logical record. A logical record key field can be the name of any field defined for the logical record.
- The field is a table column in a CA Culprit run that accesses one or more tables. *Key-field-name* must be the same as the column specified in the WHERE clause on the INPUT parameter.

database-record-name* OF *logical-record-name

Is required under the following conditions:

- If *key-field-name* occurs in more than one database record within a logical record.
- If *key-field-name* references a column that occurs more than once in a table view.

If specified, this expression must be enclosed in parentheses.

When the key field references a column in a table view, *database-record-name* exists in the form RFUR-*nnnnnn*-DATA where *nnnnnn* is the table definition number (TDN) of the source table. The TDN appears on ASF's Table Definition screen. It also appears as the numeric portion of the subschema value assigned to the source table in CA Culprit's Input Parameter Listing.

In addition, *logical-record-name* is the logical record name assigned to the table view. The name appears in ASF's Extended Table Definition screen. It also appears in the LR-NAME field in CA Culprit's Input Parameter Listing.

key-value-expression

See expanded syntax shown below:

key-value

Identifies a key value or a list of key values for entry record occurrences. Each value in a list must be separated by a space or a comma and the list must be enclosed in parentheses. Key values must be either CALC, index, logical record, or table column keys. When *key-value* refers to a logical record or a table column, CA Culprit automatically substitutes the value of *key-value* for KEY-VALUE on the PATH or INPUT parameter.

Key-value is either an alphanumeric, numeric, or hexadecimal value, depending on the definition of *key-field-name*:

- An **alphanumeric** value is a 1- to 64-character literal enclosed in single quotation marks. If the length of the literal (excluding the quotation marks) is longer than the field length defined for *key-field-name*, CA Culprit truncates excess characters on the right. If the length of the literal is shorter, CA Culprit pads the literal on the right with blanks.
- A **numeric** value is defined as either a 1- to 31-digit literal if the entry record is a database record, or a 1- to 15-digit literal if the entry record is a logical record. The literal can contain a trailing or an embedded decimal point but not a leading decimal point.

  CA Culprit automatically converts *key-value* to the numeric field format for *key-field-name* in the subschema. If the number of decimal places specified for *key-field-name* in the subschema differs from the number specified for the numeric literal, CA Culprit truncates or expands the literal value to correspond with the subschema definition. If the number of digits specified for the numeric literal is more than the number defined for *key-field-name* in the subschema, CA Culprit truncates the numeric literal on the left. If the number of digits is less, CA Culprit pads the literal value on the left with zeros.
- A **hexadecimal** value is preceded by an X and is enclosed in single quotation marks; for example, X '00104B1A'. The length of the literal must contain exactly the same number of bytes defined for *key-field-name* in the subschema.

start-key-value

Specifies the starting value for a range of indexed records. Coding specifications for *start-key-value* are the same as for *key-value*.

FIRST

A keyword that specifies the first value in the indexed set.

end-key-value

Specifies the ending value for a range of indexed records. Coding specifications for *end-key-value* are the same as for *key-value*.

LAST

A keyword that specifies the last value in the indexed set.

The range must be enclosed in parentheses and the values that specify the range must be separated by the keyword TO. The values specified for the start key and end key do not have to exist in the database; the values only serve to establish a range within the index.

DBKEY

A keyword indicating that database key values are used to access database entry records. DBKEY can be specified for database entry records stored in any location mode (that is, CALC, VIA, or DIRECT), but cannot be specified for an indexed database entry record or for a logical record.

database-key

Specifies the value of the db-key. It is an 8-character hexadecimal literal enclosed in single quotation marks and preceded by an X; for example, X'00104B1A'. The first six digits are the CA IDMS/DB page number of the database entry key; the last two digits are the line number. The hexadecimal value must include any leading zeros.

page-number/line-number

Specifies two numeric literals that, respectively, represent the page number and line number of the database key. The values must be separated by either a blank space or a comma.

Note: Users with schemas that employ a floating radix point must specify db-key values as hexadecimal literals.

Usage

The following considerations apply to the KEY parameter:

- Any number of KEY parameters can be entered in a CA Culprit run.
- KEY parameters should be coded after the INPUT parameter; otherwise, CA Culprit issues a warning (W-level) message.
- If more than one KEY parameter is entered, each KEY parameter must specify the same key field.

Examples

Sample KEY parameters are shown and described below.

Example 1: KEY Parameter with CALC-key Values

```
PATHAA DEPARTMENT EMPLOYEE
KEY DEPT-ID (0100 3200 5100)
```

The KEY parameter identifies DEPT-ID as the key field for the DEPARTMENT entry record. CA Culprit accesses only those department record occurrences that have a DEPT-ID value of 0100, 3200, or 5100. The key values must be CALC-key values because the entry record on the PATH parameter has a CALC location mode and is not followed by an index set name.

An alternative coding method appears below:

```
PATH DEPARTMENT EMPLOYEE
KEY DEPT-ID 0100
KEY DEPT-ID 3200
KEY DEPT-ID 5100
```

Example 2: KEY Parameter with Index-key Values

```
PATH EMPLOYEE(EMP-LNAME-NDX) COVERAGE
KEY  EMP-LAST-NAME (FIRST TO 'G')
```

The KEY parameter instructs CA Culprit to access all EMPLOYEE record occurrences with an EMP-LAST-NAME value within the specified range of index key values. EMP-LAST-NAME must be an index key field because the PATH parameter specifies an index set name after the entry record. The range expression, FIRST TO 'G', includes all records that begin with letters A through F, starting with the first record in the index. 'G' serves only to establish the range; it is not an actual value in the index.

Example 3: KEY Parameter with Logical Record Key Values

```
PATHAA DEPT-EMP-LR WHERE EMP-LAST-NAME EQ KEY-VALUE
KEY EMP-LAST-NAME (EMPLOYEE OF DEPT-EMP-LR) ('BREEZE' 'CRANE')
```

Occurrences of the DEPT-EMP-LR logical record are retrieved when the value of EMP-LAST-NAME in the EMPLOYEE database record of the logical record equals 'BREEZE ' or 'CRANE '. The field length of EMP-LAST-NAME is 15 bytes, so CA Culprit pads the values supplied on the KEY parameter with blanks. The information in parentheses following EMP-LAST-NAME on the KEY parameter implies that EMP-LAST-NAME is defined for more than one database record in the logical record.

Example 4: KEY Parameter and Data Table Columns

```
 INPUT TABLE=SALARY TYPE=COPY
*    WHERE DEPT-NAME EQ KEY-VALUE
 KEY DEPT-NAME ('PERSONNEL')
```

DEPT-NAME is the name of a column in table SALARY. In this example, CA Culprit accesses all rows in the table that specify 'PERSONNEL'.

Example 5: KEY Parameter with Db-key Values

```
KEY DBKEY 19035 02
KEY DBKEY 1245 36
```

CA Culprit retrieves two database record occurrences by their db-key page and line numbers. The first record occurs on page 19035, line 2, and the second is located on page 1245, line 36.

Alternatively, the db-keys can be coded as hexadecimal literals:

```
KEY DBKEY X'004A5B02'
KEY DBKEY X'0004DD24'
```

The first six digits of each literal represent the page number; the last two digits represent the line number. Both literals include leading zeros.

About the KEYFILE Parameter

Purpose

The KEYFILE parameter is an optional parameter that specifies the characteristics of a sequential file that contains key values. The key file field contains the following types of values, depending on whether the entry record is a database record, logical record, or a table:

Database Record

If the entry record is a database record, the value of the key field must be a CALC key, an index key, or a db-key:

- If the entry record is accessed by using an index, the key values must be index keys.
- If the entry record is not accessed by using an index and has a CALC location mode, the key values can be CALC keys.
- If the entry record is not accessed by using an index and has any type of location mode, the key values can be db-keys.

The key file fields that contain CALC-key or index-key values must be defined with the same field length and data type as the field in the database; for example, if the database field contains a 2-byte binary value, the values in the key file must be 2-byte binary values.

Logical Record or Table

If the entry record is a logical record or a table, the following options can be used to select logical records or table rows:

- Key file values can be compared to the values of any logical record field or table column by using KEY-VALUE in the WHERE clause on the PATH or INPUT parameter. The KF= expression on the KEYFILE parameter designates the field in the key file. The LRFNAME= expression on the KEYFILE parameter designates the logical record field or table column. Both keyword expressions are described in the syntax rules below.
- If the KEYFILE parameter includes the FN= option, any key file field can be referenced in the WHERE clause on the PATH or INPUT parameter.

  Note: When a key file field is referenced in a WHERE clause, no KEY parameters can be specified.

Associated Parameters

Only one KEYFILE parameter can be specified for a CA Culprit run. Parameters that are associated with the KEYFILE parameter follow:

REC Parameters

User-supplied REC parameters define fields in the sequential file unless the file is defined to the data dictionary, in which case the file must be named in the FN= expression on the KEYFILE parameter. User-supplied REC parameters must be qualified by the keyword KEYFILE.

SELECT/BYPASS Parameters

SELECT/BYPASS parameters can be specified to define selection criteria to be applied to each record in the sequential file.

SELECT/BYPASS parameters associated with the KEYFILE parameter must appear in the input stream after the KEYFILE parameter and before any subsequent INPUT or PATH parameters. SELECT/BYPASS parameters associated with a KEYFILE parameter are coded in the same way as SELECT/BYPASS parameters associated with INPUT parameters.

More information:

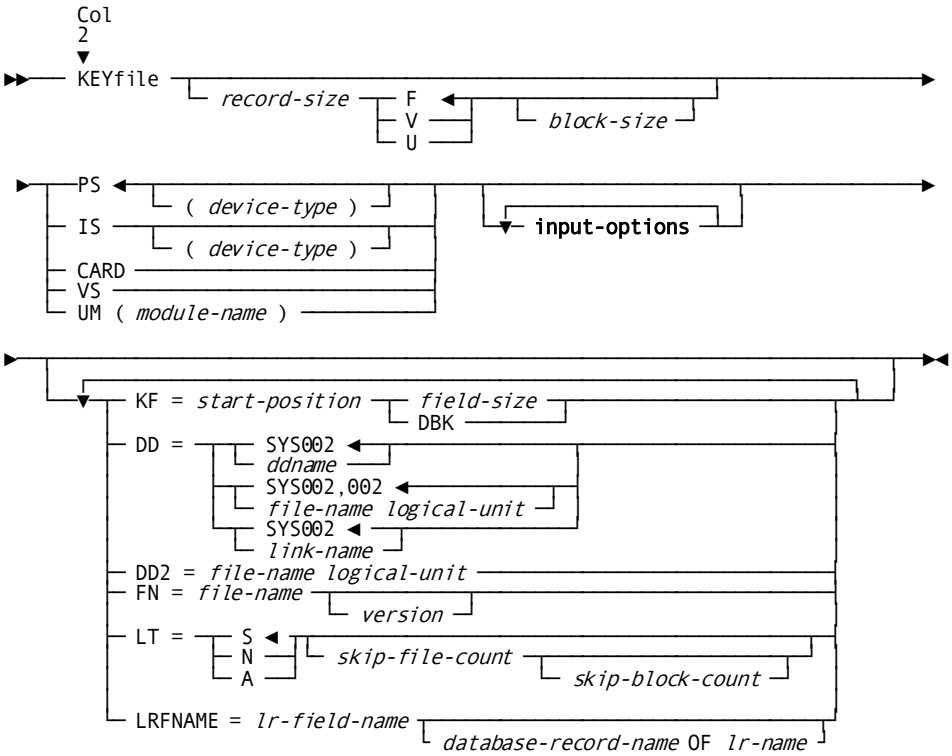
REC Parameter (see page 214)
SELECT / BYPASS Parameters (see page 60)

KEYFILE Parameter

Purpose

Specifies the characteristics of a sequential file that contains key values.

Syntax



Syntax Rules

Syntax rules for the KEYFILE parameter are the same as those described for the INPUT parameter in Chapter 3, Input Definition Parameters, with the exception of special KEYFILE characteristics described below:

KEYFILE

Identifies the parameter type; it must be coded starting in column 2.

KF =

A keyword expression that indicates the location and size of the key field in the sequential file. This keyword expression must be specified under the following circumstances:

- When the entry record on the PATH parameter is a database record
- When the entry record on the PATH parameter is a logical record and KEY-VALUE is used in the WHERE clause
- In a CA Culprit run that accesses tables and KEY-VALUE is used in the WHERE clause

start-position

Specifies the starting position of the key field in the key file record.

field-size

Specifies the length, in bytes, of the key field, if it is not a db-key field. *Field-size* must be the same value as the length of the field in the database record, logical record, or a table.

DBK

A keyword that identifies *start-position* as the beginning of a db-key. The key file must contain db-key values in an 8-digit hexadecimal format. Leading zeros, if any, must be included.

DD =

Specifies the ddname (z/OS or z/VM), filename (z/VSE), or linkname for the key file.

SYS002/*ddname*

(z/OS and z/VM only). Specifies the name assigned to the DD or FILEDEF statement that defines the key file; the default is SYS002.

SYS002,002/*file-name logical-unit*

(z/VSE only) Specifies the name assigned to the input file in execution JCL and the number of the device on which the appropriate file resides; the default is SYS002,002.

Logical-unit must be a number in the range 0 through 255. When DD=*file-name* is specified, *logical-unit* must also be specified. The default logical unit number is 002.

FN =

Specifies that the key file is defined to the data dictionary.

file-name

The name of the sequential file defined to the Integrated Data Dictionary (IDD). If the name contains special characters (that is, anything other than letters, numbers, or hyphens), it must be enclosed in single quotation marks.

version

Optionally identifies the version of the file to be accessed. If *version* is omitted, CA Culprit accesses the highest version number for the named file.

When the KEYFILE parameter specifies a FN= keyword expression, IDD automatically supplies CA Culprit with the characteristics used to define the file to the data dictionary. If appropriate values are set in the data dictionary, the user can omit the following file characteristics from the KEYFILE parameter: the record size, record type, block size, file type, device type, or user module name or label type.

IDD also automatically generates a REC parameter for each key file field referenced on other CA Culprit parameters. Depending upon the security features in effect, the user may or may not be able to redefine a key file field by coding a user-supplied REC parameter.

Note: If the WHERE clause references a key file field, the field must be defined to the data dictionary; that is, it cannot be a user-defined field.

LRFNAME =

Identifies either a logical record field in the CA Culprit input buffer or a table column name. This keyword expression is required only when the WHERE clause on the PATH or INPUT parameter specifies KEY-VALUE.

For each record in the key file, CA Culprit places the key value into the input buffer field identified by the LRFNAME= keyword expression. References to KEY-VALUE in the WHERE clause then access the value in the input buffer field. Each logical record that meets the criteria specified in the WHERE clause is processed. CA Culprit resets the input buffer value before it retrieves the next logical record occurrence in case the value was altered by the previous DBA-specified retrieval path.

lr-field-name

Is either the name of a field in the logical record or a table column name. The size and data type of the field must be the same as that defined for the key file field. For example, if the logical record field contains a 2-byte binary value, the values in the key file must be 2-byte binary values.

database-record-name* OF *lr-name

Is required if *lr-field-name*:

- Occurs in more than one database record within a logical record.
- Is the name of more than one column in a table view.

If specified, this expression must be enclosed in parentheses.

When the field name is the name of a column in a table view, *database-record-name* exists in the form RFUR-*nnnnnn*-DATA, where *nnnnnn* is the table definition number (TDN) of the source table. The TDN appears on ASF's Table Definition screen. It also appears as the numeric portion of the subschema value assigned to the source table in CA Culprit's Input Parameter Listing.

In addition, *lr-name* is the logical record name assigned to the table view. The name appears in ASF's Extended Table Definition screen. It also appears in the LR-NAME field in CA Culprit's Input Parameter Listing.

Examples

Example 1: KEYFILE Parameter and CALC-key Values

```
PATHAA DEPARTMENT EMPLOYEE
KEYFILE 80 KF=10 4
```

The sequential file defined by the KEYFILE parameter contains 80-byte fixed-length records. By default, the file is identified on DD statement SYS002. The key file field begins in position 10 for 4 bytes. The key values must be CALC-key values because the entry record on the PATH parameter is a database record that is not accessed by using an index.

Example 2: KEYFILE Parameter and Db-key Values

```
PATH01 EMPLOYEE DEPARTMENT
PATH02 EMPLOYEE OFFICE
KEYFILE 150 F 300 PS(TAPE) KF=5 DBK
```

The sequential tape file contains 150-byte fixed-length records; each block contains two records. The file contains the db-key values for database record occurrences of the EMPLOYEE record; each db-key value is an 8-character hexadecimal (4-byte) value that starts in position 5 of the key file record. CA Culprit retrieves all EMPLOYEE records that have associated db-key values and related DEPARTMENT and OFFICE records.

Example 3: KEYFILE Parameter and Associated REC and SELECT Parameters

```
PATH01 EMPLOYEE DEPARTMENT
PATH02 EMPLOYEE OFFICE
KEYFILE 150 F 300 PS(TAPE) KF=5 DBK
REC EMP-ID (KEYFILE)   20 4
SELECT EMP-ID (KEYFILE) EQ (0001 TO 0099)
```

This example is similar to the one above except that CA Culprit selects key file records that have a value for EMP-ID in the range 1 through 99. The REC parameter identifies EMP-ID as a field in a keyfile. The SELECT parameter applies the selection criteria to the key file values. After the key file records are selected, CA Culprit accesses only those record occurrences that have db-keys equal to the key file values.

Example 4: KEYFILE Parameter and Automatic File Definition

```
KEYFILE FN=EMPL-FILE KF=23 10
```

In this example, the sequential key file is fully defined to the data dictionary. IDD supplies CA Culprit with the characteristics of the file, such as the record length and record type. The 10-byte field that contains the key values starts in position 23 of each key file record.

Example 5: KEYFILE Parameter and Logical Record Key Values

```
PATH01 DEPT-EMP-LR WHERE EMP-LAST-NAME EQ KEY-VALUE
KEYFILE CARD KF=10 15 LRFNAME=EMP-LAST-NAME
```

The key values in this card file begin in position 10 of each card for 15 bytes. CA Culprit compares the key file values to the value of EMP-LAST-NAME in the logical record. If the values match, CA Culprit retrieves all DEPT-EMP-LR logical records that specify the key file value.

Example 6: KEYFILE Parameters and Logical Record Key Values

```
 PATH03 DEPT-EMP-LR WHERE EMP-NAME EQ KEY-EMP-NAME
*                   AND DEPT-ID EQ 5100
 KEYFILE FN=KEY-EMP-FILE
```

KEY-EMP-FILE is the name of the key file defined to the data dictionary. KEY-EMP-NAME, which appears on the PATH parameter, is a field in the key file. CA Culprit retrieves all logical records where the value for the logical record field EMP-NAME equals the value of the key file field and where the value of the department ID in the logical record is 5100.

Example 7: KEYFILE Parameter and Data Table Columns

```
 INPUT TABLE=PERSONNEL TYPE=COPY
*      WHERE EMP-ID-0415 EQ KEY-VALUE
 KEYFILE 80  KF=1 4 LRFNAME=EMP-ID-0415
```

The key file contains employee ids starting in position 1 for a length of four bytes. CA Culprit accesses all rows in table PERSONNEL where the value in column EMP-ID-0415 is the same as the ID in the key file field.

More information:

CA Culprit Security Considerations (see page 192)

SELECT / BYPASS Parameters

More information:

SELECT / BYPASS—Overview (see page 59)

What They Do

The SELECT/BYPASS parameters specify selection criteria that are applied to each input record; if the input record meets the selection criteria, it is processed in the CA Culprit run.

SELECT/BYPASS parameters apply selection criteria to different record types, as follows:

- SELECT/BYPASS parameters select records in a sequential file that is defined on the KEYFILE parameter.
- SELECT/BYPASS parameters limit unnecessary database record retrieval. If a database record occurrence does not meet the selection criteria specified on the SELECT/BYPASS parameters, CA Culprit does not continue to navigate the path defined on the PATH parameter.
- SELECT/BYPASS parameters select rows from a table. For example, if a table contains rows of information for all departments in a company, the SELECT or BYPASS parameter selects rows of information to be processed for specified departments.
- SELECT/BYPASS parameters apply selection criteria to the completed input buffer.

More than one SELECT or BYPASS parameter can be coded for each entity; however, both SELECT and BYPASS parameters cannot be applied to the same parameter or buffer.

Since SELECT or BYPASS parameters can apply selection criteria to different parameter types and to the input buffer, placement of SELECT/BYPASS parameters in the input stream is significant.

Considerations

- SELECT/BYPASS parameters should immediately follow their associated KEYFILE or PATH parameters.
- If SELECT/BYPASS parameters do not immediately follow their associated KEYFILE and PATH parameters in a database access run, SELECT/BYPASS selection criteria is applied to all preceding PATH parameters or to the KEYFILE parameter last encountered.
- SELECT/BYPASS parameters that specify the BUFFER option must follow any INPUT, KEYFILE, and PATH parameters specified for the CA Culprit run.

Listings

The Input Parameter Listing and Run Time Message listing provide information about CA Culprit's interpretation of selection criteria and the actual number of records tested. The following figure illustrates both listings. The Input Parameter Listing assigns a reference number to each selection operation. The reference number appears again in Run Time Messages under the Selection Specification Statistics, which indicate how many records were tested and how many passed or failed the test. The CA IDMS/DB Database Extract Statistics in the same listing indicate how many strings were returned for a specific path and how many of these strings were not processed due to selection criteria.

```
mm/dd/yy          INPUT PARAMETER LISTING      Vnn.n        PAGE    1

********
SEL/BYP  REF       CONDITION
********
 SEL            $     DEPARTMENT IN PATH AA WHEN DEPT-ID-0410 EQ (5100 3200)  ┐
 SEL    00001      DEPT-ID-0410        EQ        5100                          │
 SEL    00002      DEPT-ID-0410        EQ        3200                          ├─ CA-Culprit's interpretation
 SEL            $     EMPLOYEE IN PATH DD WHEN EMP-ID-0415 EQ (0301 0054 0074) │  of selection criteria
 SEL    00003      EMP-ID-0415         EQ        0301                          │
 SEL    00004      EMP-ID-0415         EQ        0054                          │
 SEL    00005      EMP-ID-0415         EQ        0074                          ┘
mm/dd/yy          RUN TIME MESSAGES          Vnn.n        PAGE     1
  IDMS DATABASE EXTRACT STATISTICS                    ┐
  STRINGS RETURNED FOR PATH AA -          15          ├─ Number of strings CA-Culprit
  STRINGS TRUNCATED BY SELECTION CRITERIA          24  │  returned for each path and
  STRINGS RETURNED FOR PATH BB -          24          │  bypassed due to selection
  STRINGS RETURNED FOR PATH CC -           0          │  criteria
  STRINGS RETURNED FOR PATH DD -           5          │
  STRINGS TRUNCATED BY SELECTION CRITERIA          83 ┘
  STRINGS RETURNED FOR PATH EE -           0
  RECORD NAME       NUMBER READ
 DEPARTMENT                 26
 EMPLOYEE                   85
 OFFICE                     15
 COVERAGE                    5
**********  END OF FILE  **************************************************
         44 INPUT RECORDS READ
        SELECTION SPECIFICATION STATISTICS
        BEGINNING      TIMES      TIMES      TIMES    NUMERIC  SUBSCRIPT  ┐
        REFERENCE      TESTED      TRUE      FALSE     ERRORS     ERRORS   │
         NUMBER                                                           ├─ Number of strings tested by
                                                                          │  the selection criteria
            1           26          2         24          0          0    │
            3           85          2         83          0          0    ┘
```

Note: The Input Parameter Listing interprets and assigns a reference number to each SELECT/BYPASS statement. The Run Time Messages listing indicates how many strings were processed for each path and the number of times each SELECT/BYPASS statement was tested.

More information:

CA Culprit in the CA IDMS/DB Environment (see page 187)
About the KEYFILE Parameter (see page 257)

SELECT/BYPASS parameters

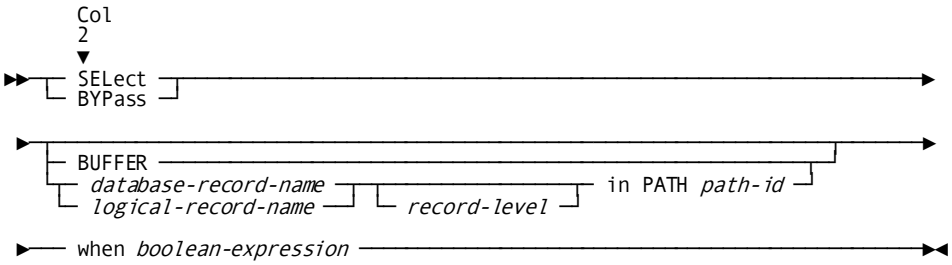
More information:

PATH Parameter (see page 220)
SELECT / BYPASS Parameters (see page 60)
Database Field Name References (see page 244)

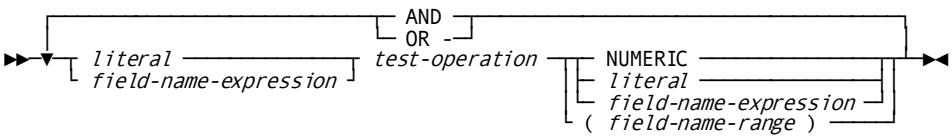
Purpose

Specify selection criteria that are applied to each input record.

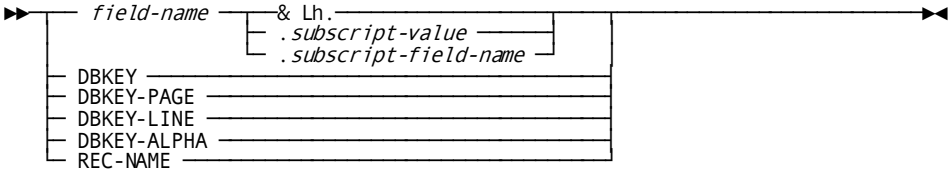
Syntax



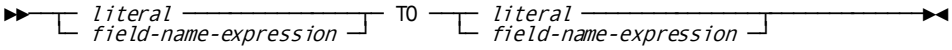
Expansion of boolean-expression



Expansion of field-name-expression



Expansion of Field-name-range



Syntax Rules

Except as described below, syntax rules for SELECT/BYPASS parameters are the same as those described under SELECT/BYPASS Parameters (see page 60). Special syntax rules that apply to database access follow:

database-record-name/logical-record-name

Specifies the name of either a database record, logical record, or a table in a preceding PATH parameter:

- For a database record or logical record, the SELECT/BYPASS selection criteria are applied to fields in the named record or in any records previously named in the specified path.
- For a CA Culprit run that accesses tables, the SELECT/BYPASS selection criteria are applied to each table. The SELECT/BYPASS parameters must specify the logical record name of the primary table.

CA Culprit evaluates multiple SELECT/BYPASS parameters in the following manner:

- If the parameters specify the same record name for the same PATH parameter, CA Culprit evaluates SELECT or BYPASS parameters as if they were connected by OR. In the following example, CA Culprit selects DEPARTMENT record occurrences that either have an ID of 5100 or 3200 or that specify BRAINSTORMING for a name:

```
PATHAA DEPARTMENT EMPLOYEE
SELECT DEPARTMENT WHEN DEPT-ID-0410 EQ (5100 3200)
SELECT DEPARTMENT WHEN DEPT-NAME-0410 EQ 'BRAINSTORMING'
```

- If the parameters specify different record names for the same PATH parameter, CA Culprit evaluates each SELECT or BYPASS parameter as if they were connected by AND; that is, if the first record accessed fails the selection criteria, CA Culprit does not access subsequent records on the path. Therefore, the selection criteria for a later record are not applied. In the example shown below, CA Culprit processes only those strings for which the department ID is 5100 and the employee last name is CRANE:

```
PATHAA DEPARTMENT EMPLOYEE
SELECT DEPARTMENT WHEN DEPT-ID-0410 EQ 5100
SELECT EMPLOYEE WHEN EMP-LAST-NAME-0415 EQ 'CRANE'
```

record-level

Applies when the named record occurs more than once on a single path. *Record-level* is an integer in the range 1 through 256 that specifies the occurrence level of the named record; the default is 1.

IN PATH *path-id*

Specifies that the selection criteria apply to the named record on a particular path; *path-id* must be a primary path id. If no path ID is specified, the selection criteria apply to all preceding PATH parameters that specify the named record.

boolean-expression

CA Culprit tests this expression at run time when it retrieves a record or fills an input buffer. See the expanded syntax described below.

Syntax rules for this expression are described in Chapter 3, Input Definition Parameters. The following considerations apply to database access:

- When CA Culprit retrieves a record occurrence, the selection criteria are applied just once for all paths that specify the record name, ensuring maximum efficiency. For example, CA Culprit applies the selection criteria for the EMPLOYEE record before it returns either path AA or path BB to the input buffer:

  ```
  PATHAA DEPARTMENT EMPLOYEE EXPERTISE
  PATHBB DEPARTMENT EMPLOYEE COVERAGE
  SELECT EMPLOYEE WHEN EMP-ID-0415 EQ 0054
  ```

  Therefore, the selection criteria that are applied to path AA cannot be changed before they are applied to path BB by changing the value of a global work field within the procedure logic of a report, as shown below:

  ```
  GW0  ID   0054
  SELECT EMPLOYEE WHEN EMP-ID-0415 EQ ID
  ```

- When *field-name-expression* is a database record field name, the field must exist in the record identified by *database-record-name* or in any previously named records on the specified path or paths. In the example shown below, DEPT-ID-0410 must be a field in either the EMPLOYEE or DEPARTMENT record; it cannot be a field in the EMPOSITION record:

  ```
  PATHAA DEPARTMENT EMPLOYEE EMPOSITION
  SELECT EMPLOYEE WHEN DEPT-ID-0410 EQ 4000
  ```

- When SELECT/BYPASS parameters are associated with a KEYFILE parameter, *field-name-expression* must be the name of a field in a key file.
- The following CA Culprit keywords and associated record and level qualifiers can replace *field-name-expression* in the syntax presented above:
  - DBKEY
  - DBKEY-PAGE
  - DBKEY-LINE
  - DBKEY-ALPHA
  - REC-NAME

Examples

Sample SELECT/BYPASS parameters are shown and described below.

Example 1

```
PATH01 EMPLOYEE COVERAGE
SELECT EMPLOYEE IN PATH 01 WHEN EMP-ID-0415 LT 0100
```

CA Culprit selects all EMPLOYEE records in path 01 that have an employee ID value less than 100. CA Culprit will not retrieve the COVERAGE records for employees with an ID that exceeds this limit, and will not return a string for path 01.

Example 2

```
PATH01 EMPLOYEE DEPARTMENT
PATH02 EMPLOYEE COVERAGE
SELECT EMPLOYEE WHEN EMP-ID-0415 LT 0100
```

This example is similar to Example 1, except that the SELECT parameter does not specify a path id. In this case, CA Culprit applies the selection criteria to both path 01 and path 02.

Example 3

```
PATH01 EMPLOYEE(02) DEPARTMENT
SELECT EMPLOYEE WHEN EMP-ID-0415 LT 0100
```

For each employee record with an employee ID less than 100, CA Culprit returns a string consisting of the EMPLOYEE record and the DEPARTMENT record; the path ID is 01. For all other employee records, CA Culprit returns a string that contains the EMPLOYEE record; the path ID is 02.

Example 4

```
PATHB7 DEPARTMENT EMPLOYEE EXPERTISE
SELECT EMPLOYEE WHEN EMP-ID-0415 EQ (0048 0053 0074)
SELECT EMPLOYEE WHEN DEPT-ID-0410 EQ 4000
```

Of the strings returned to the input buffer for path B7, CA Culprit selects only those strings that specify either the employee ID or department ID values coded on the SELECT parameters.

Example 5

```
PATHB8 DEPARTMENT EMPLOYEE EXPERTISE
SELECT EMPLOYEE WHEN EMP-ID-0415 EQ (0048 0053 0074)
SELECT DEPARTMENT WHEN DEPT-ID-0410 EQ 4000
```

This example differs from the preceding example because the two SELECT parameters specify different database record names. Of the strings returned for path B8, CA Culprit selects only those strings that specify both 4000 for the department ID and 0048, 0053, or 0074 for the employee id.

Example 6

```
KEYFILE 80 KF=5 4
REC EMP-ID (KEYFILE) 5 4 2
BYPASS EMP-ID EQ (0479 0329 0015)
```

The KEYFILE parameter describes the characteristics of a sequential file that contains employee ID key values. The BYPASS parameter instructs CA Culprit to ignore all key file records with employee ID values equal to the values in parentheses.

Example 7

```
INPUT TABLE=ACCOUNTING TYPE=COPY
INPUT TABLE=PERSONNEL TYPE=CONSOL
SELECT ACCOUNTING WHEN EMP-NAME-0415 GT 'M'
```

In this example, CA Culprit consolidates the tables ACCOUNTING and PERSONNEL. The SELECT parameter processes only those rows in both tables where the employee's name begins with the letters in the range N through Z.

OUTPUT Parameter

More information:

OUTPUT Parameter (see page 70)

Overview

The OUTPUT parameter specifies how each report in the CA Culprit run is to be output. Part I of this manual described output directed to printers, sequential files, card files, and ISAM files.

This chapter describes output that creates and updates tables.

Sample Programs—Table Access and Creation (see page 403) presents examples of CA Culprit code that creates and updates tables.

OUTPUT Parameter

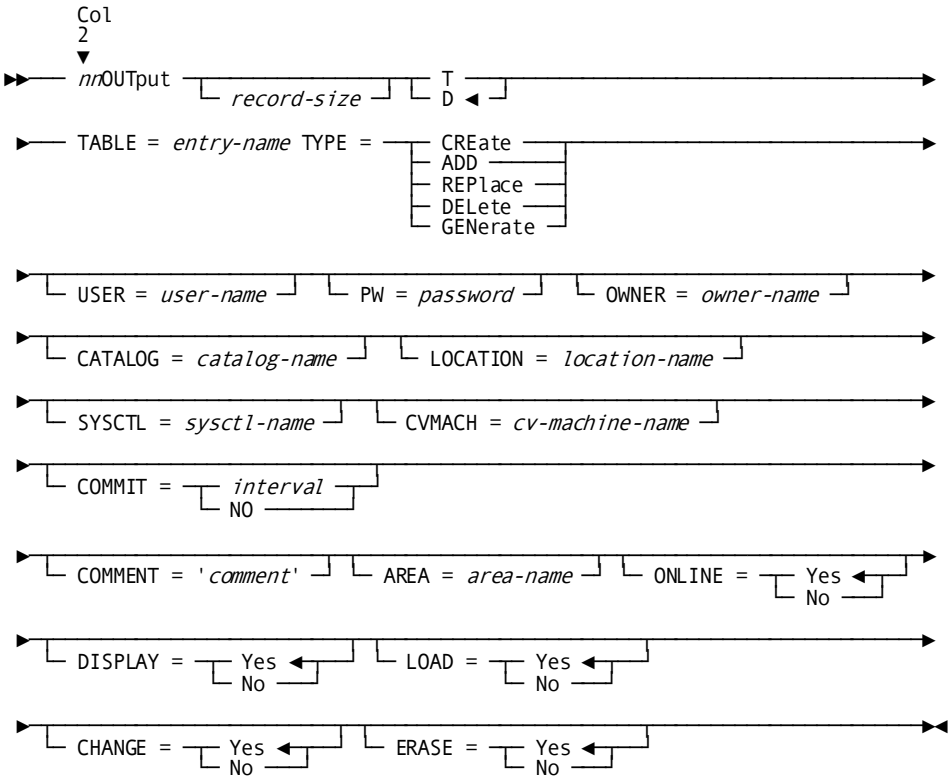
More information:

OUTPUT Parameter (see page 70)

Purpose

Specifies how each report in the CA Culprit run is to be output.

Syntax



Syntax Rules

Syntax rules follow:

nn

Identifies the report associated with the OUTPUT parameter. It is a 2-digit number in the range 00 through 99 and must be coded starting in column 2.

OUTPUT

Specifies the parameter type. It must be coded starting in column 4.

record-size

Specifies the length of the rows in the table. It is an integer in the range 1 through 32,760. Specify the record size for documentation purposes only; the default is the record size of the table.

T/D

Indicates a totals-only (T) or details-only (D) report. The default is D.

TABLE = *entry-name*

Names the table. TABLE *must* be the first keyword on the OUTPUT parameter. *Entry-name* is a 1- to 64-character value. If the name contains embedded blanks or special characters, it must be enclosed in single or double quotation marks. If enclosed in quotation marks, the entire name must appear on a single line. Otherwise, the name can span several lines:

```
 01OUTPUT TABLE=MZC.        $Owner.Folder.Object
*               REPORT01    $Note that the period
*               .BUDGET     $can appear on either line
```

TYPE =

Specifies keywords to create or update tables. TYPE= must be the second keyword expression on the OUTPUT parameter.

CREATE

Defines, generates, and populates a new table. When a table is created, it is defined automatically to the data dictionary.

Note: A table gets defined to the data dictionary in the CULL step of processing, even if the CA Culprit job contains run-time errors.

ADD

Adds rows to an existing table. The edit parameters coded for the report must specify the same length, format, and order as those defined for the original table.

REPLACE

Erases previously stored data and stores new data. The edit parameters coded for the report must specify the same length, format, and order as those defined for the original table.

DELETE

Deletes a stored table and all associated data. DELETE also removes the table definition from the data dictionary.

Note: A table gets deleted in the CULL step of processing, even if the CA Culprit job contains run-time errors.

To delete a table, code the OUTPUT parameter as well as a dummy INPUT parameter, REC parameter, and type 5 edit parameter to satisfy CA Culprit's coding requirements. For example, in a z/OS environment, the code appears as follows:

```
//SYS010  DD DUMMY
 INPUT 80
 REC RECORD-NAME 1 80
 01OUTPUT TABLE=PERSONNEL TYPE=DELETE
 0151*010 ' '
```

GENERATE

Regenerates a table definition that has been modified. You must regenerate a table under the following conditions:

- You have modified the column definitions by using ASF.
- You have modified the table definition (for example, the amount of space allocated to the table) by using ASF or CA Culprit.

Note: A table gets generated in the CULL step of processing, even if the CA Culprit job contains run-time errors.

To regenerate a table, code the OUTPUT parameter as well as a dummy INPUT parameter, REC parameter, and type 5 edit parameter to satisfy CA Culprit's coding requirements. For example, in a z/OS environment, the code appears as follows:

```
//SYS010  DD DUMMY
//SYSIN   *
 INPUT 80
 REC RECORD-NAME 1 80
 01OUTPUT TABLE=PERSONNEL TYPE=GENERATE
 0151*010 ' '
```

USER = ***user-name***

Identifies the individual with authority to sign on to the catalog and overrides the user specified on the PROFILE parameter, if any. *User-name* is a 1- to 32-character value.

PW = ***password***

Indicates the user's 1- to 8-character alphanumeric password. It overrides the password specified on the PROFILE parameter.

OWNER = *owner-name*

Identifies the table owner. When the user is not the table owner, this keyword expression is required. CA Culprit determines whether the owner has assigned the user passkey authority to update the table.

CATALOG = *catalog-name*

Identifies the catalog (dictionary) containing the table definition. This keyword expression allows you to output tables to any number of catalogs. *Catalog-name* is an optional 1- to 8-character alphanumeric expression. To determine valid specifications, see your DBA.

If *catalog-name* is omitted, the default is as follows:

- CA Culprit uses the database name (if specified) in the DBNAME= keyword expression on the DATABASE parameter.
- If no database name is specified, CA Culprit uses the data dictionary name (if specified) in the DICTNAME= keyword expression on the DATABASE parameter.
- If neither a database or data dictionary name is specified on the DATABASE parameter, the default is the startup dictionary.

LOCATION = *location-name*

(DDS users only) Identifies the DDS node used to access the catalog containing the table. This keyword expression allows you to output tables to any number of nodes in a given DDS network.

Location-name is a 1- to 8-character value. The default is the value specified in the DICTNODE= keyword expression on the DATABASE parameter. To determine valid specifications, see your DBA.

SYSCTL = *sysctl-name*

(z/OS and z/VSE users only) Overrides the SYSCTL= clause on the DATABASE parameter. If there is no SYSCTL value on the DATABASE parameter, CA Culprit uses the default value in IDMSOPTI. The SYSCTL= keyword expression allows users to output tables to any number of central versions.

CVMACH = *cv-machine-name*

(z/VM users only) Overrides the value of the CVMACH= clause on the DATABASE parameter and the central version specified in an IDMSOPTI module. *Cv-machine-name* is a 2- to 8-character value that identifies the virtual machine in which the CA IDMS/DB central version system is executing. With this keyword expression, CA Culprit can output tables to any number of central versions in the same CA Culprit job step.

COMMIT =

Specifies the interval at which DML COMMIT statements are issued when a table is updated:

interval

An integer that specifies the interval at which COMMITs are to be issued. The default is 100.

COMMIT statements are issued when:

- REPLACE erases the specified number of rows and after all rows have been erased
- CREATE, ADD, or REPLACE stores the specified number of rows and after all rows for a table have been stored

NO

Indicates that no COMMIT statements are to be issued.

Note: For more information about COMMIT statements, see the *CA IDMS Navigational DML Programming Guide*.

The following keywords modify fields that appear on the ASF Table Definition screen and the Extended Table Definition screen. For more information about these screens, see the *CA IDMS ASF User Guide*.

AREA = *area-name*

(CREATE and GENERATE functions only) Names the database area in which the table's data is stored. If specified, it updates the AREA field on ASF's Extended Table Definition screen. *Area-name* must be the name of a valid area defined by the ASF administrator.

ONLINE = YES/NO

(CREATE and GENERATE functions only) Specifies whether maps and dialogs are generated when a table is created or regenerated. The default is NO. If specified, this keyword expression updates the GENERATE MAP/DIALOG field on ASF's Extended Table Definition screen.

When maps and dialogs are generated, users can display, add, and modify data in the table online by using ASF.

DISPLAY = YES/NO

(CREATE and GENERATE functions only) indicates whether ASF should generate An OBTAIN logical record path in the ASF-generated subschema. The default is YES. If specified, this keyword expression updates the DISPLAY ACCESS field on ASF's Extended Table Definition screen.

LOAD = YES/NO

(CREATE and GENERATE functions only) Indicates whether ASF should generate a STORE logical record path in the ASF-generated subschema. The default is YES. If specified, this keyword expression updates the LOAD ACCESS field on ASF's Extended Table Definition screen.

CHANGE = YES/NO

Indicates whether ASF should generate a MODIFY logical record path in the ASF-generated subschema. The default is YES. If specified, the keyword expression updates the CHANGE ACCESS field on ASF's Extended Table Definition screen.

ERASE = YES/NO

Indicates whether ASF should generate an ERASE logical record path in the ASF-generated subschema. The default is YES. If specified, this keyword expression updates the ERASE ACCESS field on ASF's Extended Table Definition screen.

Examples

Example 1: Creating a Table

```
 DATABASE DICTNAME=ASFDICT
 PROFILE USER=MZC PW=HAZELNUT
  -
  -
 34OUTPUT TABLE=EMPTABLE TYPE=CREATE ONLINE=YES
*         COMMENT='TABLE OF COMPANY EMPLOYEES'
```

Report 34 defines and populates a table EMPTABLE. By default, MZC owns the table, which is defined in ASFDICT, the dictionary named on the DATABASE parameter. Also by default, the report contains details-only information. Because the code includes ONLINE=YES, MZC can display and update the table's data online by using ASF.

Example 2: Replacing a Table

```
 DATABASE DICTNAME=DOCANWK
 PROFILE USER=PPETERS PW=PANDA
 INPUT DB SS=EMPSS01,EMPSCHM,100
 PATHAA DEPARTMENT EMPLOYEE EMPOSITION
   -
   -
   -
 34OUTPUT TABLE=BUDGET TYPE=REPLACE CATALOG=ASFDICT
*         OWNER=PMS
 3451*010 DEPT-ID-0410        FN
 3451*020 EMP-NAME-0415
 3451*030 SALARY-AMOUNT-0420  FP
```

CA Culprit accesses dictionary DOCANWK for the subschema definitions and input for the CA Culprit run. The OUTPUT parameter instructs CA Culprit to replace the existing data in table BUDGET with new information. User PPETERS must have passkey authority to access the table, which is owned by PMS.

BUDGET is defined in the ASFDICT catalog. The sequence and format of the edit parameters are the same as when the table was created.

Example 3: Adding Rows to a Table

```
 INPUT 80 F 320
 REC EMP-ID       1  4  2
 REC EMP-NAME     5 25
 01OUTPUT TABLE=EMPLOYEES TYPE=ADD
*         USER=DEH PW=CHESTNUT CATALOG=ASFDICT
 01510001 EMP-ID    FN
 01510026 EMP-NAME
```

CA Culprit reads input data from a sequential file and adds new rows of data to table EMPLOYEES. To run the job successfully, the data type and sequence of the edit parameters are the same as defined for the table.

Example 4: Deleting Tables

```
//SYS010 DD DUMMY
//SYSIN  *
 DATABASE DICTNAME=ASFDICT
 PROFILE USER=DOC1 PW=DOC1
 INPUT 80
 REC DUMMY-NAME 1 80
 01OUTPUT TABLE=EMPLOYEES TYPE=DELETE
 01510001 ' '
 02OUTPUT TABLE='PERSONNEL BUDGET' TYPE=DELETE
 02510001 ' '
```

The OUTPUT parameters instruct CA Culprit to delete two stored tables. To satisfy CA Culprit's coding requirements, the code includes a dummy INPUT parameter, REC parameter, and a type 5 edit parameter for each report. By default, the owner of both tables is DOC1 and the tables are defined in the ASFDICT catalog.

Example 5: Regenerating a Modified Table

```
//SYS010 DD DUMMY
//SYSIN  *
 INPUT 80 F 320
 REC DUMMY-NAME 1 80
 01OUTPUT TABLE=ACCOUNTING TYPE=GENERATE
*         USER=DOC1 PW=DOC1 CATALOG=ASFDICT
*         COMMENT='TABLE REGENERATED WITH CULPRIT'
 01510001 ' '
```

In this example, user DOC1 modified the column definitions of table ACCOUNTING on ASF's Column Definition screen. To regenerate the table, DOC1 codes a CA Culprit report that specifies GENERATE on the OUTPUT parameter. The COMMENT= keyword expression updates the description of the table contained in the COMMENTS field of ASF's Table Definition screen.

To satisfy CA Culprit's coding requirements, the code also includes a dummy INPUT parameter, REC parameter, and a type 5 edit parameter.

Edit Parameters

More information:

Edit Parameters — Overview (see page 82)

About Edit Parameters

Edit parameters define the position and format of fields to be output.

This chapter provides the syntax rules for edit parameters that send output to non-SQL created data tables.

For syntax rules that apply to edit parameters for reports output to a printer or conventional file structures, see Edit Parameters (see page 84).

Except as described below, the syntax rules described under Edit Parameters (see page 84) apply to edit lines defined for tables.

See Sample Programs — Table Access and Creation (see page 403) for examples of CA Culprit code that creates and modifies tables.

Edit Parameters

More information:

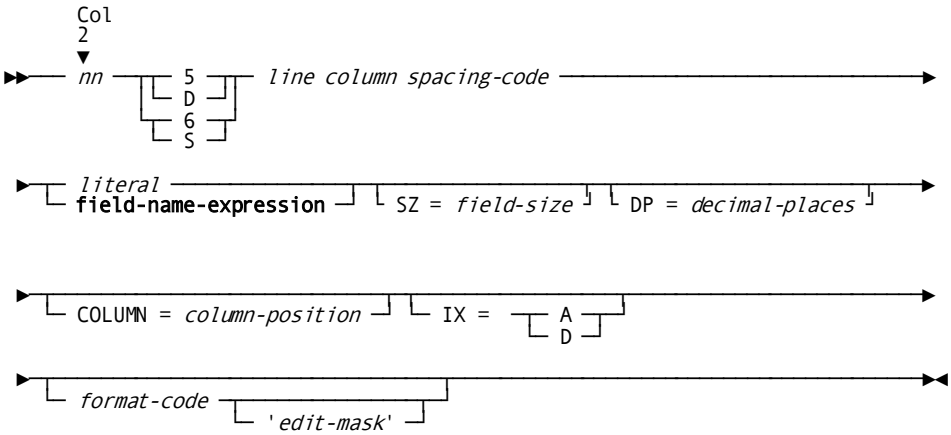
Edit Parameters — Overview (see page 82)

Purpose

Sends output to tables.

Syntax

Syntax for edit parameters is shown below:



Syntax Rules

5/6

Specifies an edit parameter that defines an output field on a detail or total line, respectively. The following coding considerations apply:

- Each report must specify at least one type 5 edit parameter.
- For details-only reports, the sequence, format, and data type of the fields specified on the type 5 edit parameters define the table's columns.
- For totals-only reports, the sequence, format, and data type of the fields specified on the type 6 edit parameters define the table's columns.
- If the OUTPUT parameter specifies ADD or REPLACE, the output fields on type 5 edit parameters must have the same length, format, and order as the fields defined for the original table; the field names do not have to be the same.
- If the OUTPUT parameter specifies DELETE or GENERATE, the report can specify a dummy type 5 edit parameter in order to satisfy Culprit's coding requirements.

line

Specifies an edit line; for output to a table, *line* must have a value of 1.

column

Specifies the location of the output field on the output line as either an absolute or a relative column position. When tables are created, the type 5 and type 6 edit parameters should specify relative columns (specified as **nnn*) because tables must have contiguous fields.

***literal*/field-name-expression**

Specifies an output field on an edit parameter. When the OUTPUT parameter specifies CREATE, CA Culprit uses the name of each field as a column heading. For example, if the field name is EMP-ID, CA Culprit automatically creates a column heading that specifies this field name.

For expanded syntax for **field-name-expression**, see Edit Parameters — Overview (see page 82).

SZ=*field-size*

Specifies the length of the output field:

- For alphanumeric fields, SZ= specifies the length in bytes.
- For numeric format codes other than FB and FP, SZ= specifies the number of decimal digits.
- For numeric fields output as packed decimal (FP) or binary (FB), SZ= specifies the length in bytes.

The default is the length defined for the field.

IX=A/D

This is a keyword expression that establishes an ascending (A) or descending (D) index for a column in the table. An index arranges the rows of a table according to the values of the indexed column. Information from an indexed table can be accessed and retrieved more efficiently than from a nonindexed table.

The IX= keyword expression can be used only if the OUTPUT parameter specifies CREATE. Only one column can be indexed per table. Noncontiguous keys cannot be concatenated.

format-code/'edit-mask'

Specifies format options for numeric output. *Edit-mask* describes the size and format of an output field that specifies format code FM. For more information about this format code, see Edit Parameters (see page 84).

The format codes listed in the following table store the data as numeric. Format codes other than the ones listed in the following table store the data as text with embedded edit characters; for example, F$ stores the numeric value 1234 as the text value $1,234. The length Culprit assigns to the table field includes the edit characters.

Format Code	Size in Bytes	Data Type
None	1-32,760	Alphanumeric
FP	1-16	Packed signed decimal
FZ	1-31 (1-18 for ASF)	Zoned decimal
FB	2 or 4	Binary
FU	4	Floating point - single precision
FW	8	Floating point - double precision

Examples

Sample edit parameters for detail and total lines are shown and described below.

Example 1

```
 010UTPUT D TABLE=TABLE1 TYPE=CREATE USER=LHN PW=WALNUT
*          CATALOG=ASFDICT
 0151*010 EMP-NAME   SZ=25 IX=A
 0151*020 SALARY     SZ=10 F$  DP=2
```

The edit parameters for Report 01 define two columns of information. The first column has EMP-NAME as the column heading and lists employee names. CA Culprit indexes this column in ascending alphabetic order by the value of EMP-NAME. The second column has SALARY as the column heading and lists the salaries associated with each employee. Because the SALARY field specifies format code F$, CA Culprit stores the values as text with a floating dollar sign and decimal point; arithmetic operations cannot be performed on the values.

Example 2

```
 01OUTPUT D TABLE=TABLE1 TYPE=ADD USER=DAV PW=PEANUT
*           CATALOG=ASFDICT
 0151*010 NAME        SZ=25
 0151*020 SALARY      SZ=10  F$  DP=2
```

This example is similar to Example 1, except that the OUTPUT parameter specifies ADD. The order, field size, and formats for the output fields are the same as those specified in Example 1. However, the field name in relative column *010 specifies NAME rather than EMP-NAME. CA Culprit can still verify this field because it compares the field size and data type of each type 5 edit parameter, starting with the lowest column position, to each table column definition. It does not compare the column names.

Example 3

```
 01OUTPUT TABLE=TABLE1 TYPE=DELETE USER=JFD PW=ACORN
*        CATALOG=ASFDICT
 0151*010 ' '
```

Since the OUTPUT parameter specifies DELETE, Report 01 specifies a dummy type 5 edit parameter to satisfy Culprit's coding requirements.

Example 4

```
 DATABASE DICTNAME=TSTDICT
 INPUT DB SS=EMPSS01,EMPSCHM,100
 REC DEPT-ID(DEPARTMENT)  1  4  2
 REC SALARY(EMPOSITION)  15  5  3
 PATHAA DEPARTMENT EMPLOYEE EMPOSITION
 01OUT T TABLE='DEPARTMENT SALARIES' TYPE=CREATE USER=DOC1 PW=DOC1
*        CATALOG=ASFDICT ONLINE=YES COMMENT='CREATED 2/18/87'
 01SORT DEPT-ID +
 0151*020 SALARY
 0161*010 DEPT-ID   FN
 0161*020 SALARY    FP
```

In this example of a totals-only report, type 6 edit parameters define the columns for table DEPARTMENT SALARIES and provide the column names. Each total line will contain the following information:

- A department ID defined as a text field because the edit parameter specifies format code FN
- The total employee salary for each department defined as a packed decimal numeric field

Database-Specific Process Parameters

Overview

Process parameters define process functions that are to be performed on data before it is output to a report.

Process parameter	What it does
Type 7	Specifies a process operation that is to be performed on input data.
Type 8	Specifies a process operation that is to be performed on system-maintained totals of extracted items.

All of the database access considerations that appear in this chapter apply to type 7 process parameters only.

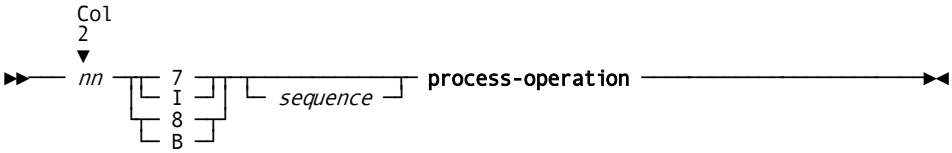
Syntax for the process parameter is shown below.

Process Parameters

Purpose

Defines process functions that are to be performed on data before it is output to a report.

Syntax



Syntax Rules

Syntax rules for Process parameters are presented in Process Parameters (see page 103) chapter.

process-operations

See expanded syntax later in this chapter.

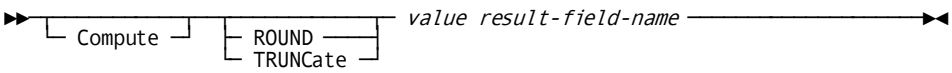
Except as described below, syntax rules for *process-operation* in this chapter apply to database access.

Arithmetic Operations

Purpose

Arithmetic operations perform addition, subtraction, multiplication, and division.

Syntax



Usage

The following reserved words can serve as operands in an arithmetic statement:

- DBKEY
- DBKEY-PAGE
- DBKEY-LINE

More information

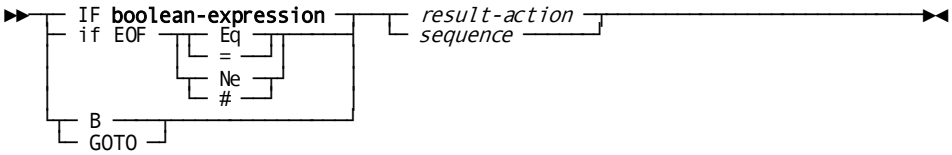
Database Field Name References (see page 244)

Conditional Operations

Purpose

Conditional operations define test conditions that compare values held by different fields.

Syntax



Syntax Rules

boolean-expression

For expanded syntax see SELECT / BYPASS — Overview (see page 59).

For database access, the following reserved words can replace **field-name-expression** in the syntax:

PATH-ID	DBKEY-LINE
REC-NAME	DBKEY-ALPHA
IDMS-STATUS	TABLE-ID
LR-STATUS	TABLE-NAME
DBKEY	SUBSCHEMA-NAME
DBKEY-PAGE	

result-action

For database access, DB-EXIT is a possible value for *result-action* except in the case of an end-of-file (EOF) test operation. DB-EXIT instructs CA Culprit to pass processing control to CA IDMS/DB; CA IDMS/DB retrieves a requested database record and returns it to the CA Culprit input buffer.

More information:

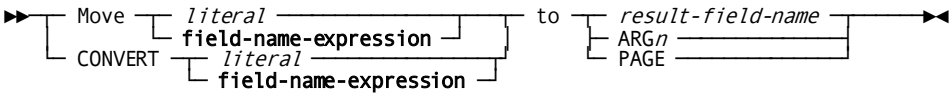
Database Field Name References (see page 244)
The DB-EXIT Facility (see page 429)

Assignment Operations

Purpose

Assignment operations assign a value contained in a literal or variable field to either a work field or a CA Culprit reserved field.

Syntax



Syntax Rules

MOVE

Specifies that a MOVE operation is to be performed.

field-name-expression

With a MOVE, the following reserved words can replace **field-name-expression** in the syntax:

PATH-ID	DBKEY-LINE
REC-NAME	DBKEY-ALPHA
IDMS-STATUS	TABLE-ID
LR-STATUS	TABLE-NAME
DBKEY	SUBSCHEMA-NAME
DBKEY-PAGE	

CONVERT

Specifies that a CONVERT operation is to be performed.

field-name-expression

With CONVERT, the following reserved words can replace **field-name-expression** in the syntax:

- PATH-ID
- REC-NAME
- IDMS-STATUS
- LR-STATUS
- DBKEY-ALPHA

Used in MOVE operations to pass arguments to the DB-EXIT facility, which is described in detail in Appendix L, The DB-EXIT Facility.

ARG*n* is a reserved word, where *n* is an integer in the range 1 through 9.

More information:

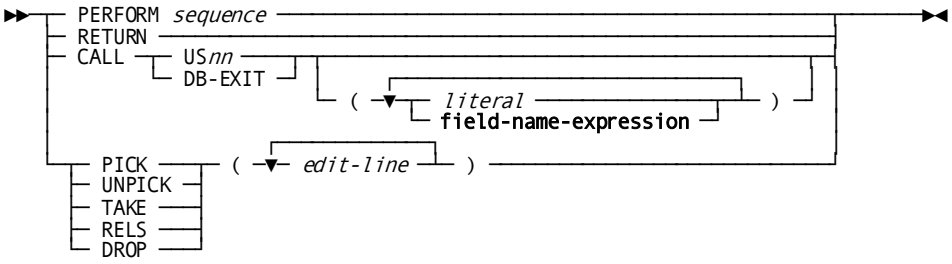
Database Field Name References (see page 244)

Control Operations

Purpose

These control the flow of processing in a CA Culprit report.

Syntax



Syntax Rules

For a discussion of the basic syntax rules, see Control Operations (see page 123).

For use with a database, the following special syntax rules apply:

CALL DB-EXIT

Instructs CA Culprit to pass processing control to the DB-EXIT facility, which is described in detail in Appendix L, The DB-EXIT Facility.

***literal*/field-name-expression**

Specifies literals, input fields, or work fields whose values are used as arguments for the DB-EXIT call. CA Culprit moves each value to an ARG*n* field.

As many as nine arguments can be specified. The argument values must be specified in sequence; that is, ARG1 corresponds to the first value in the list, ARG2 corresponds to the second argument in the list, and so on.

Argument values for each type of DB-EXIT call appear in Appendix L, The DB-EXIT Facility.

Examples

Sample process parameters are shown and described below.

Example 1

```
017010 PATH-ID EQ 'AA' TAKE
```

CA Culprit compares the value of the PATH-ID field in the input buffer to AA. If the comparison is true, CA Culprit extracts all the type 5 edit parameters coded for Report 01.

Example 2

```
017010 MOVE 'OWNER'            TO ARG1
017020 MOVE 'NON-HOSP-CLAIM '  TO ARG2
017030 MOVE 'COVERAGE-CLAIMS ' TO ARG3
017040 IF REC-NAME EQ 'NON-HOSP-CLAIM' DB-EXIT
```

The MOVE operations transfer literal values to arguments that are associated with a branch to DB-EXIT. CA Culprit branches to DB-EXIT if the value of REC-NAME in the input buffer equals the database record named in process statement 040.

Example 3

```
017010 IF REC-NAME EQ 'NON-HOSP-CLAIM' 500
              -
              -
017500 CALL DB-EXIT ('OWNER' 'NON-HOSP-CLAIM ' 'COVERAGE-CLAIMS ')
```

This example illustrates an alternate method to execute a branch to DB-EXIT. CA Culprit compares the value of REC-NAME in the input buffer to the literal coded on process statement 010. If the comparison is true, CA Culprit branches to process statement 500 and calls the DB-EXIT facility, using the literals within the parentheses as values for ARG1, ARG2, and ARG3, respectively.

Chapter 10: Accessing SQL Defined Tables

This section contains the following topics:

Retrieving Data with SQL

CA Culprit can retrieve data from CA IDMS/DB tables using Structured Query Language (SQL). A new SQL parameter has been added to CA Culprit syntax that allows the user to code a complete SQL select statement This select statement is used at two different times during CA Culprit processing:

- During the **Precompile Phase** the SQL commands *Prepare* and *Describe* are issued to provide CA Culprit with an SQL Descriptor area (SQLDA) that describes the format of the data that is retrieved. This information is used by CA Culprit to automatically generate REC parameters to describe the format of the CA Culprit input buffer.
- During the **Extract Phase** the SQL SELECT clause is prepared once again, and data is retrieved from CA IDMS/DB using the highly efficient *bulk fetch* technique. All cursor manipulation is completely automated by CA Culprit, so that data is returned, one row at a time, to the CA Culprit input buffer.

Coding Considerations

Very few syntax changes are needed to use the SQL retrieval feature. Each CA Culprit parameter affected is discussed below.

DATABASE

The DATABASE parameter is optional. If it is present, it must always be the first parameter in your CA Culprit syntax.

PROFILE

The PROFILE parameter is also optional. The USER= and PW= clauses should be specified if CA Culprit security is on.

INPUT

The INPUT parameter identifies the input data as a CA IDMS/DB SQL table. New syntax for the INPUT parameter is given later in this chapter.

SQL

The new SQL parameter immediately follows the INPUT parameter. Here, the user codes an SQL SELECT clause that is passed along to the database engine. The SELECT clause may span multiple lines, providing the CA Culprit continuation character (*) appears in column 1. Comments are not allowed in the SQL statement.

REC

REC parameters are automatically generated to describe both the SQL columns and the null value indicators that is returned to the CA Culprit input buffer. As always, the user can code additional REC parameters to either re-define columns or to define a "dummy buffer". Further discussion of generated REC cards appears later in this chapter.

SELECT/BYPASS

The SELECT/BYPASS parameter is completely functional with SQL retrieval. SELECT/BYPASS gives you a Selection Statistics Report that tells you the number of rows selected, the number of rows rejected, and the total number of rows on the input table. However, you can select rows far more efficiently by coding a WHERE clause on your SQL SELECT statement. This latter technique is strongly recommended whenever you don't need the selection statistics information.

OUTPUT

Data retrieved from SQL tables can be written to any supported output media. Thus CA Culprit can be used as a migration tool, to quickly convert SQL tables into another data format.

EDIT

There are no changes for coding type 4, 5, and 6 parameters. Use the field names from the generated REC cards to reference SQL columns and their null value indicators.

PROCESS

There are no changes for coding type 7 and 8 parameters. Use the field names from the generated REC cards to reference SQL columns and their null value indicators.

Your detail time logic should check null indicators whenever appropriate. When an SQL column is null, the null indicator has a value of minus one, and the value in the SQL column is unpredictable.

JCL Considerations

SQL retrieval can be done either in local mode or CV mode. For the best CA Culprit performance, use local mode.

Either one-step or five-step JCL can be used. If running with five-step JCL, provide database DD statements for PGM=CULP0 and PGM=CULL.

Creating New SQL Tables

Introduction

CA Culprit can create new CA IDMS/DB tables. The user simply codes familiar CA Culprit syntax, and SQL commands are generated automatically. Minimal training in Structured Query Language is needed for a CA Culprit programmer to begin creating new SQL tables.

Creating new SQL tables is a two step process: Create the table definition, and insert new data.

- During the **Compile Phase** of CA Culprit, information from the OUTPUT parameter and the EDIT parameters is gathered, and SQL Create Table syntax is generated. The syntax is then passed to the CA IDMS/DB database engine, and a new SQL table definition is created. A message appears on the Input Parameter Listing telling the user that the Create Table procedure is successful, and the data extraction phase begins.
- During the **Output Phase** the bulk insert technique is used to load data into the SQL table. CA Culprit automates this procedure completely.

Coding Considerations

DATABASE

The DATABASE parameter is optional. If it is present, it must always be the first parameter in your CA Culprit syntax.

PROFILE

The PROFILE parameter is also optional. The USER= and PW= clauses should be specified if CA Culprit security is on.

INPUT

Input data can be retrieved from any accessible source: Sequential files, ISAM, VSAM, IMS, DL/I, CA IDMS/DB non-SQL records, ASF tables, CA IDMS/DB SQL tables, and so on. Data retrieval is independent from SQL table creation. Thus, CA Culprit can be used as a migration tool, reading data from one medium and converting it to an SQL table.

OUTPUT

Up to 100 reports can be processed in a single CA Culprit run. Any number of these reports can update SQL tables, using the create, add, replace, and drop functions. Data written to SQL tables can be details-only or totals-only.

DETAIL

DETAIL (Type 5) parameters describe the columns and null indicators for the detail-only SQL table being created. If a totals-only table is being created, type 5 parameters are used to specify automatic subtotals for numeric fields.

TOTAL

TOTAL (Type 6) parameters are only used when creating totals-only SQL tables. They describe the columns and null indicators for the SQL table being created.

PROCESS

Type 7 parameters let the user code processing logic that is executed after each input record is read. Type 8 logic is executed when each control break is encountered at total-time. No changes to these process parameters are needed when writing output to SQL tables.

JCL Considerations

SQL updates take place in CV mode, which automatically provides full database recovery. Local mode updates are possible if the central version is down and the areas are unlocked. Consult your database administrator to establish local mode backup and recovery procedures before attempting local mode SQL updates.

Either one-step or five-step CA Culprit JCL can be used. If running five-step, then SYSCTL and SYSIDMS statements are required for PGM=CULL and PGM=CULE.

Adding and Replacing Data on SQL Tables

Introduction

In addition to creating new CA IDMS/DB tables, CA Culprit can also add and replace data on existing SQL tables. Many of the coding techniques discussed earlier in this chapter also apply to adding and replacing data.

Updating SQL tables is a two step process:

1. Validating the output data buffer layout against the existing table definition,
2. Inserting new data.

- During the **Compile Phase** of CA Culprit, an SQL Prepare/Describe is executed for *SELECT * FROM table-name*. This returns an SQL Definition Area (SQLDA) defining all columns in the table. CA Culprit uses this information to validate the edit parameters coded by the user.
  - Info-column numbers are used to sequence the edit parameters coded by the user.
  - For each SQL column, the SQL data type must match the CA Culprit data specification. Field names are not considered.
  - An edit parameter with data specification *FB SZ=4* must follow every SQL column that may contain null values. This numeric field represents the null indicator for the previous SQL column. During the output phase, the value of this field must be either zero (indicating not null) or minus one (indicating a null value).
- During the *Output Phase*, CA Culprit begins by issuing the SQL command *DELETE FROM table-name* if TYPE=REPLACE was coded. Next, use the highly efficient *bulk insert* technique to load data into the SQL table. All SQL processing is handled automatically by CA Culprit.

Coding Considerations

Everything previously discussed for DATABASE, PROFILE, INPUT, OUTPUT, and PROCESS parameters also applies for the add and replace functions. Differences arise in the way EDIT parameters are processed.

For TYPE=CREATE, the EDIT parameters specified the SQL column definitions that are generated. The field name becomes the new SQL column name; the sequence of EDIT parameters define the SQL column sequence; and the CA Culprit data specification becomes the SQL data type.

For TYPE=ADD and TYPE=REPLACE, the EDIT parameters are compared against an existing SQL table definition and validated. *E* level errors result if the CA Culprit data specification differs from the known SQL data type.

JCL Considerations

CV mode is recommended for the ADD and REPLACE functions. CV automatically provides full database recovery. Local mode updates are possible if the central version is down and the areas are unlocked. Consult your database administrator to establish local mode backup and recovery procedures before attempting local mode SQL updates.

You can use either one-step or five-step CA Culprit JCL. If running five-step, then SYSCTL and SYSIDMS statements are required for PGM=CULL and PGM=CULE.

Dropping SQL Tables

Introduction

The purpose of the drop table function is simply to drop an existing SQL table definition and delete its data. This takes place during the compile phase of CA Culprit. For a report of TYPE=DROP, the extract phase and the output phase are not really necessary.

Coding Considerations

The following CA Culprit parameters are used to drop SQL tables.

INPUT

An INPUT statement is always required for a CA Culprit job, even though it is irrelevant for the drop table function.

REC

A REC parameter is also required for every CA Culprit job. You define a single REC parameter when just drop table is being performed.

nnOUTPUT

The OUTPUT parameter is where you specify the SQLTABLE= clause and TYPE=DROP. You also specify the SQL dictionary and schema name on the OUTPUT parameter.

A complete discussion of the OUTPUT syntax is found in OUTPUT Parameter (see page 300).

The clause CASCADE=YES is optional for TYPE=DROP. CASCADE instructs CA IDMS/DB to delete the definition of:

- All referential constraints in which the table is either the referencing table or the referenced table.
- All views derived from the named table.

If the table you are trying to delete is involved in any linked referential constraints, and the related table is not empty, CA Culprit is not able to drop this table. You should then use the Command Facility to delete the data from both tables, drop the linked constraints, and then drop the table.

Note: For more information about the Command Facility, see the *CA IDMS Common Facilities Guide*.

Type 5

A type 5 parameter is required in every CA Culprit report, even though you do not want to print any output.

Type 7

To prevent data extraction, you code a single type 7 parameter to DROP the input record.

Drop Table Example

```
 col. 2
 ▼
 IN 80
 REC FIELDA   14
 01OUTPUT  SQLTABLE=INVENTORY     TYPE=DROP
*          DICTIONARY=TSTDICT     SCHEMA=INV
 0151*001 ' '
 017    DROP
```

JCL Considerations

Drop Table reports should be run in CV mode.

Either one-step or five-step CA Culprit JCL can be used. If running five-step, then SYSCTL and SYSIDMS statements are required for PGM=CULL.

INPUT Parameter

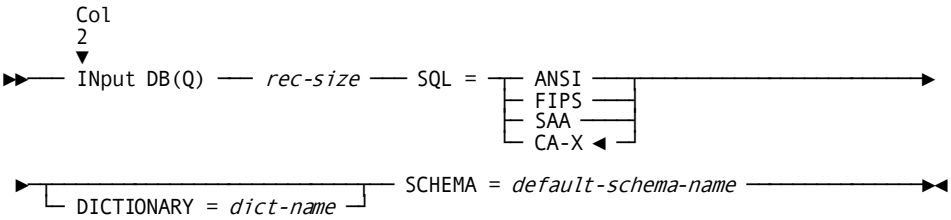
More information:

INPUT Parameter (see page 43)

Purpose

The INPUT parameter describes the input medium as a CA IDMS/DB SQL table. It also identifies the SQL dictionary and schema name.

Syntax



Syntax Rules

INput DB(Q)

Identifies this parameter as an INPUT statement.

rec-size

Allocates storage for the CA Culprit input buffer. If omitted, CA Culprit allocates the larger of 1000 bytes or the row length as determined by the SQL descriptor area.

DB(Q)

Identifies the source of input as a CA IDMS/DB SQL table.

DICTIONARY=

Specifies the name of the SQL dictionary. The name can be up to eight characters. Single quotes are optional.

SCHEMA=

Specifies the name of the default schema for this session. The name can be up to 18 characters. Single quotes are optional.

SQL=

Identifies the dialect of SQL that the SELECT clause is written in. Valid SQL options include CAX, ANSI, FIPS, and SAA. CAX is the default and represents compliance with CA IDMS/DB Extended SQL.

BULKROWS=

Determines the number of rows retrieved during each bulk fetch operation. For greater performance, try increasing this value. If omitted, CA Culprit builds a fetch buffer for 100 rows of data.

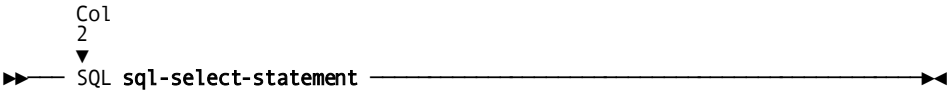
SQL Parameter

Purpose

The SQL parameter contains the SELECT clause that is passed to the CA IDMS/DB database engine to perform REC card generation and data retrieval.

Note: For more information on the SELECT statement syntax, see the *CA IDMS SQL Reference Guide*.

Syntax



Syntax Rules

SQL

Identifies the parameter. Must begin in column 2.

sql-select-statement

Any SQL SELECT statement, according to the syntax rules of the selected compliance.

Note: For more information on authorization requirements, expanded syntax, parameter descriptions, usage notes and examples, see the *CA IDMS SQL Reference Guide*.

Usage

Reminders

- SQL must begin in column 2
- The SQL parameter must immediately follow the INPUT parameter
- Continuation lines must have an asterisk in column 1
- No comments are allowed in the SQL statement

Use the column names from the SQL defined data table when coding Edit parameters (types 4, 5, and 6).

The SQL statement can be continued across any number of lines using the standard CA Culprit continuations character (*) in column 1. At the detection of the last continued line, the entire SQL SELECT statement is passed unchanged to the database engine to generate REC parameters.

Note that SQL comments can NOT be placed into the SELECT statement.

Example

```
 SQL SELECT "EMP-ID-0415"         AS  ID,
*           "EMP-LAST-NAME-0415"  AS "LNAME"
*    FROM   EMPLOYEE
*    WHERE  "EMP-ID-0415"  IN (1000,2000,3000)
*    ORDER  BY ID;
```

REC Parameter

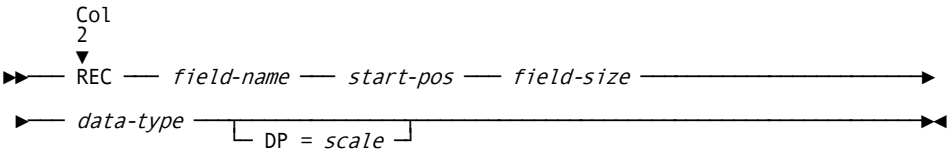
More information:

REC Parameter—Overview (see page 51)

Purpose

REC parameters are automatically generated in the precompile phase to describe each SQL column retrieved in the CA Culprit input buffer. Additional REC parameters are also generated for null indicators, whenever appropriate.

Syntax



Syntax Rules

field-name

The *field-name* for generated REC cards is the alias name, if one was assigned using an AS clause. Otherwise, the field name is simply the SQL column name.

For generated null indicator REC cards, the field-name from the previous REC card is used along with the suffix **_NULL_IND**

start-pos

Represents the starting position of each field within the CA Culprit input buffer. The first SQL column starts at position 49. Positions 1-48 are reserved for internal use.

field-size

Is the size in bytes of the SQL column. For generated null indicators, the size is always 4.

data-type

Is the CA Culprit data type code. A blank represents alphanumeric data. For generated null indicators, the data type is always 1 (binary).

The table below describes the correspondence between CA IDMS/DB and CA Culprit data representations. CA IDMS/DB has many new data types that are not yet fully supported by CA Culprit. These unsupported data types are indicated by an asterisk in the table, and are treated as alphanumeric fields. Procedure exits and other type 7 programming logic can be used to process these data values during the extract phase: (Error: Don't know what to do with column width of "25," (3).(Error: Don't know what to do with column width of "8," (3).(Error: Don't know what to do with column width of "8," (3).

CA IDMS/DB data type	Size	Type	DP=
CHAR(15)	15		
NUMERIC(5,2)	5	2	2
UNSIGNED NUMERIC(5,2)	5	2	2
DECIMAL(5,2)	3	3	2
UNSIGNED DECIMAL(5,2)	3	3	2
INTEGER	4	1	
SMALLINT	2	1	
LONGINT	8	1	
FLOAT *	4 or 8		
REAL *	4		
DOUBLE PRECISION *	8		
VARCHAR(10) *	12		
DATE *	10		
GRAPHIC(5) *	10		
VARGRAPHIC(5) *	12		
BINARY(5) *	5		

DP=*scale*

Represents the number of decimal places that are implied for a zoned or packed decimal number.

REC Values During EXTRACT

During the extract phase, rows from the SQL table are returned to the CA Culprit input buffer, one by one.

- If SQL column "ABC" is not null, the ABC_NULL_IND field is set to zero, and the ABC field contains the actual data value.
- If ABC is null, the value of ABC_NULL_IND is set to minus one. The value in the ABC field is set to zero or blanks.

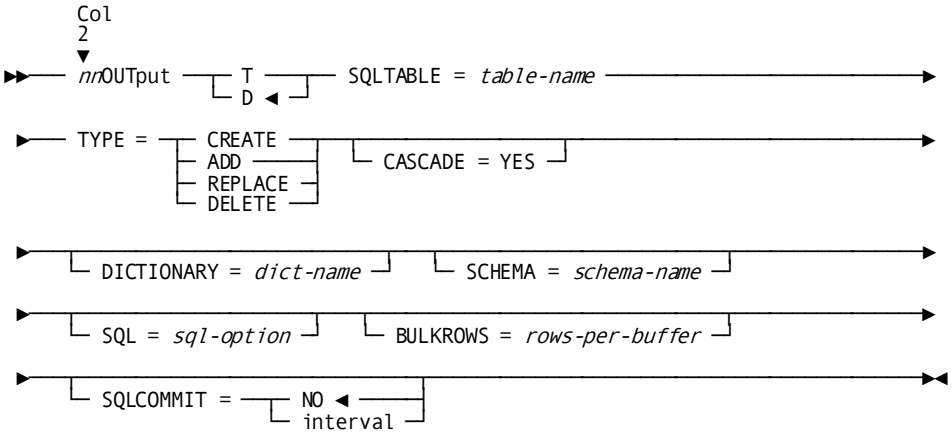
OUTPUT Parameter

More information:

OUTPUT Parameter (see page 70)

The OUTPUT parameter describes the output medium as a CA IDMS/DB SQL table. It also identifies the SQL dictionary and schema name.

Syntax



Syntax Rules

***nn*OUTput**

Defines the OUTPUT specifications for report number *nn*. The report number must begin in column 2.

T / D

T specifies that totals-only are written to the output table.

D indicates that details-only are written to the output table. Details-only is the default.

SQLTABLE= *table-name*

Specifies the name of the SQL table to be modified. The table name can be up to 18 characters long, and single quotes are optional.

TYPE=

CA Culprit can perform the following functions for output SQL tables:

Option	What it does
CREATE	Issues create table SQL syntax and inserts new rows.
ADD	Validates an existing SQL table, and inserts additional rows.
REPLACE	Validates an existing SQL table, deletes all existing rows from the table, and then inserts new rows.
DROP	Issues SQL drop table syntax to delete a table definition and all of its rows.

CASCADE= YES

This keyword only applies to the TYPE=DROP function. It directs the database to also delete the definitions of all referential constraints and all views derived from the named table.

DICTIONARY=

Specifies the name of the SQL dictionary. The name can be up to eight characters. Single quotes are optional.

SCHEMA=

Specifies the name of the default schema for this session. The name can be up to 18 characters. Single quotes are optional.

When creating new SQL tables using CA Culprit, you must specify a schema that has been previously defined to the SQL dictionary and has a default area for storing data rows.

BULKROWS=

Determines the number of rows inserted during each bulk insert operation. For greater performance, try increasing this value. If omitted, CA Culprit builds an insert buffer for 100 rows of data.

SQLCOMMIT =

Specifies the interval at which SQL COMMIT statements are issued when a table is updated.

NO

(Default) Indicates that no COMMIT statements are to be issued until all rows are updated.

interval

An integer that specifies the interval at which COMMITs are to be issued.

COMMIT statements are issued when CREATE, ADD, or REPLACE stores the specified number of rows and after all rows for a table have been stored.

Note: If the SQLCOMMIT parameter is set lower than the BULKROWS parameter, the COMMIT is issued after "BULKROWS" number of records have been stored. For more information about COMMIT statements, see the *CA IDMS SQL Programming Guide*.

EDIT Parameter

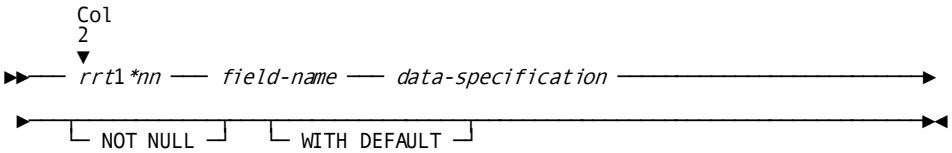
More information:

Edit Parameters — Overview (see page 82)

Purpose

For details-only tables, each type 5 detail parameter defines either an SQL column on the table, or the null indicator for the previously defined SQL column. For totals-only tables, the SQL column definitions and the corresponding null indicators are defined using type 6 total parameters.

Syntax



Syntax Rules

rrt1*nnn

Defines an edit parameter for report number *rr*.

The parameter type *t* is 5 for details-only reports and 6 for totals-only reports. Line 1 is always used when writing to SQL tables.

The info-column technique *(*nnn)* specifies the relative position of this data in the output buffer. CA Culprit automatically calculates the exact output position.

field-name

The *field-name* used on the edit parameter is used to generate SQL Create Table syntax. This field name becomes the new SQL column name. Therefore, field-names (not literals) must always be used on edit parameters when TYPE=CREATE.

data-specification

Data-specification includes the CA Culprit format code, the SZ= clause, and the DP= clause. Together, this information is used to generate the data type clause of an SQL column definition. The correspondence between the CA Culprit data specification and the CA IDMS/DB data type is shown in the following table:

CA Culprit data spec	CA IDMS/DB data type
SZ=15	CHAR(15)
FB SZ=2	SMALLINT
FB SZ=4	INTEGER
FB SZ=8	LONGINT
FP SZ=3	DECIMAL(5)
FP SZ=5 DP=2	DECIMAL(9,2)
FU	REAL
FW	DOUBLE PRECISION
FZ SZ=4	NUMERIC(4)
FZ SZ=7 DP=2	NUMERIC(7,2)

Use of the format codes FC, FD, FF, FM, FN, FS, F-, F$, F*, and F0 through F9 all produce SQL columns of data type CHAR. Data written to the SQL table is edited exactly as requested, but it is considered alphanumeric data. For most applications, this is not desirable.

CA Culprit does not currently support all of the CA IDMS/DB data types. Therefore, CA Culprit cannot create SQL tables that contain these SQL data types:

- BINARY
- BIT
- DATETIME
- FLOAT
- GRAPHIC
- UNSIGNED DECIMAL
- UNSIGNED NUMERIC
- VARCHAR
- VARGRAPHIC

NOT NULL

NOT NULL indicates that this SQL column cannot contain null values. NOT NULL is only valid when TYPE=CREATE has been specified on the OUTPUT parameter. If NOT NULL is omitted when defining an SQL column, then a null indicator field must be specified on the subsequent edit parameter.

```
0151*010  PRODUCT_CODE            SZ=4  NOT NULL
0151*020  UNIT_SALES           FP SZ=5
0151*021  UNIT_SALES_NULL_IND  FB SZ=4
```

WITH DEFAULT

WITH DEFAULT clause is only valid when NOT NULL has already been coded and TYPE=CREATE. This clause has no effect on batch data insertion. However, it simplifies online data insertion by allowing a default value to be inserted if this column is omitted from the SQL Insert statement.

Appendix A: Printing Parameter Lists

This section contains the following topics:

About This Appendix

This appendix provides information on how to control printing of the parameter lists that are supplied at run time. Two lists of CA Culprit parameters are printed at run time.

Sequential Parameter Listing

This is generated during the precompile phase of CA Culprit processing. The parameters are printed in the order they are submitted; parameters that CA Culprit generates are not shown.

The Sequential Parameter Listing also shows the parameters that are copied from stored code; the parameters for code copied by the USE parameter and by =COPY or =MACRO appear as follows:

- Code copied by the USE parameter is identified by a nesting level number in the Sequential Parameter Listing. Modifications made by CA Culprit appear in this listing.
- Code copied by =COPY and =MACRO parameters is identified by a plus sign (+) next to each copied parameter. Modifications made by CA Culprit do not appear in this listing.

Input Parameter Listing

This listing is generated during the compile phase of CA Culprit processing. The parameters are listed in an order determined by CA Culprit's standard sort. Both user-defined and CA Culprit generated parameters appear in this listing along with default values assumed by CA Culprit.

The Input Parameter Listing also shows the internal code generated by CA Culprit in response to the following conditions:

- A compound arithmetic operation identified by the keyword COMPUTE on a process definition parameter
- A compound conditional operation identified by the keyword IF on a process definition parameter
- A compound conditional operation on a SELECT/BYPASS parameter, identified by the keywords AND and OR

The internal code documents simple statements generated by CA Culprit from the compound user-coded statements. This facility allows users to follow CA Culprit's processing logic.

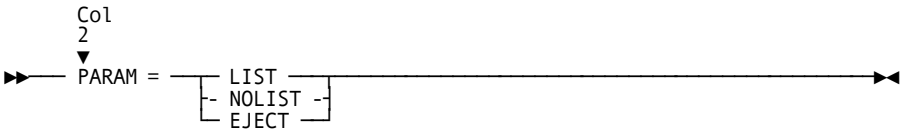
The Input Parameter Listing also includes the modifications performed by CA Culprit on code copied by an =COPY or =MACRO parameter. Modifications performed by CA Culprit on code copied by a USE parameter appear in the Sequential Parameter Listing.

PARAM

Purpose

Controls the printing of the Sequential Parameter Listing and Input Parameter Listing

Syntax



Parameters

LIST

Specifies that parameters following this statement are output in the Sequential Parameter Listing and in the Input Parameter Listing. LIST is the default.

NOLIST

Specifies that parameters following this statement are not output in either the Sequential Parameter Listing or the Input Parameter Listing, unless a parameter contains an error. If an error occurs on a parameter that follows the PARAM=NOLIST keyword expression, the parameter in error appears in the Sequential Parameter Listing and in the Input Parameter Listing with the associated error message.

EJECT

Specifies that parameters following this statement are printed in the Sequential Parameter Listing after skipping to a new page. This option does not effect the pagination of the Input Parameter Listing.

Usage

The PARAM= keyword expression can be inserted anywhere in the input parameter stream; it can be coded in any column from 2 through 72.

PARAM= statements are printed in place within the Sequential Parameter Listing; if the keyword expression specifies NOLIST, no parameters are printed after the PARAM= statement. PARAM= statements do not appear in the Input Parameter Listing.

Example

The first figure (out of the following figures) shows an example of PARAM= keyword expressions coded among input parameters. The next figure shows the Sequential Parameter Listing that results. The Input Parameter Listing appears in the final example.

Input Parameters and PARAM=NOLIST, LIST, and EJECT Options

```
IN  80
PARAM=NOLIST
REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
REC EMP- LNAME  11 15
REC SALARY     70 10 3 DP=2
PARAM=LIST
REC TITLE      50 20         'TITLE'
REC DEPARTMENT 30 20         'DEPARTMENT'
01SORT DEPARTMENT,-,TITLE
013EMPLOYEE SALARY
PARAM=EJECT
0151*010 TITLE
0151*020 EMP-NAME
```

Sequential Parameter Listing of Input Parameters:

```
mm/dd/yy                                        SEQUENTIAL PARAMETER LISTING                                  Vnn.n  PAGE  1
00 ** SYSIN **                          IN  80
  INSTALLATION SECURITY OPTION IS NO
                                        PARAM=NOLIST
                                        REC EMP- LNAME  11 15  ┐
E NON-NUMERIC PARAMETER                         *****          ├── E-level errors override the
E POSITION INVALID                              *****          │   NOLIST option for this parameter
E INVALID TOKEN IGNORED                                  **    ┘
                                        PARAM=LIST
                                        REC TITLE      50 20         'TITLE'
                                        REC DEPARTMENT 30 20         'DEPARTMENT'
                                        01SORT DEPARTMENT,-,TITLE
                                        013EMPLOYEE SALARY                    ┐
mm/dd/yy                                        SEQUENTIAL PARAMETER LISTING  ├ New page of the Sequential   Vnn.n  PAGE  2
                                        PARAM=EJECT                           ┘ Parameter Listing
                                        0151*010 TITLE
                                        0151*020 EMP-NAME
```

Input Parameter Listing of Input Parameters:

```
mm/dd/yy                                          INPUT PARAMETER LISTING                                     Vnn.n  PAGE  1
 ********
 INPUT    RECORD TYPE BLOCK  FILE DESCRIPTION...
 ********
  INPUT   00080   F
 ********
 REC      START SIZE TYPE DP FIELD-NAME                   RECORD-NAME,LEVEL
 ********
  REC     00030  020         DEPARTMENT  ┐                  'DEPARTMENT'
  REC            011         EMP-        │
 E E-LEVEL ERROR DETECTED IN CULP0 STEP  ├──────────────────────────────── E-level errors override the
 E DUPLICATE OR INVALID FIELD POSITION   ┘                                 NOLIST option for this parameter
  REC     00050  020         TITLE                          'TITLE'
 ********
 SORT     BREAK A/D  SORT FIELD-NAME
 ********
  01 SORT   -        DEPARTMENT
                     TITLE
 ********
 TITLE    REPORT TITLE
 ********
  01  3   EMPLOYEE SALARY
 ********
 EDIT     LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
 ********
  01  5     1      *010   TITLE
  01  5     1      *020   EMP-NAME
 F E-LEVEL ERRORS - EXTRACT NOT PERFORMED
  RECORDS WRITTEN FOR REPORT 01-        0
```

Appendix B: Output Phase Field References

This section contains the following topics:

Differences from Extract Phase

CA Culprit interprets field references made during the output phase of processing differently from field references made during the extract phase of processing, as follows:

- References to numeric fields can refer to column totals and values of a sort key, as well as to values obtained from process operations performed in type 8 procedure logic.
- If a subscripted field is referenced on a SORT or type 5 edit parameter, the subscript is considered part of the field name in the output phase. Otherwise, the subscript is interpreted as a pointer to an occurrence of the multiply-occurring work field. In the extract phase, the subscript always acts as a pointer to a particular occurrence of a multiply-occurring field.

Actions During Extract Phase

During the extract phase, a TAKE or RELS instruction in type 7 procedure logic causes the following types of records to be written to the extracted items and statistics file:

- **Detail records**, which contain a composite sort key and unedited data. These records are used by type 5 edit parameters to generate the detail lines for the reports.
- **HEAD records**, which contain a composite sort key and the current values of fields that are not referenced on the SORT parameter but are referenced on a type 4 edit parameter. These records are generated when HEAD is a specified result action in type 7 procedure logic processed during the extract phase.

Actions During Output Phase

The output phase reads the sorted extracted items file and the unsorted extracted items file. During the output phase, CA Culprit performs the following tasks:

- Report headings, if any, are output at the start of the report, following a control break defined with break code 1 (signifying a new page) or after the lines-per-page count is reached.
- Totals are maintained for each numeric field referenced on a type 5 edit parameter.
- Values from HEAD records are saved for use in header lines.
- Control breaks execute when a sort key changes, beginning with the lowest level control break (that is, LEVL=1) and ending with the break associated with the current value of LEVL.

  The following functions are performed at each control break:

  - LEVL is set to the appropriate value.
  - Processing control passes to type 8 procedure logic.
  - Total lines defined by type 6 edit parameters or automatic total lines are output.
  - Spacing associated with the current break code is performed.

Field references during the output phase fall into two categories: field references made on type 4 edit parameters, and field references made on type 6 edit parameters and on type 8 process parameters. The manner in which these references are resolved during the output phase is described as follows.

Field References Made on Type 4 Edit Parameters

The value of a field referenced on a type 4 edit parameter is resolved according to whether the field also appears on the SORT parameter, as follows:

- If the field appears on the SORT parameter, any reference to the field on a type 4 edit parameter yields the next value of the sort key as of the most recent control break (if any) or yields the initial value of the sort key if a control break has not occurred.
- If the field does not appear on the SORT parameter, any reference to the field on a type 4 edit parameter yields the value of the field contained in the HEAD record most recently read by the output phase. If a HEAD record has not been read by the output phase, the field appears as blanks in the header line.

Field References Made on Type 6 and Type 8 Parameters

The value of fields referenced during output phase processing on type 6 edit and type 8 process parameters depends on the following conditions:

- Whether the field is defined as a numeric or alphanumeric type 0 work field, type 1 work field, or input field
- Whether the field appears on a type 5 edit parameter
- Whether the field appears on a SORT parameter

The value of a field referenced during the output phase under the above conditions will be one of the following types of values:

- A column total
- A sort-key value
- A current value of a type 1 work field
- A current value of a type 0 work field

The following table summarizes values assigned to input and work fields during the output phase.

Filed Type		**Parameter Type**			
		SORT	**SORT and Type 5**	**Type 5**	**Other**
REC	A	Sort-key value	Sort-key value	Error	Error
REC	N	Sort-key value	Column total	Column total	Error
Work	A	Sort-key value	Sort-key value	Current value	Current value
Work	N	Sort-key value	Column total	Column total	Current value

Notes:

N = Numeric field
A = Alphanumeric field
Sort-key value = Value of sort key at the control break
Column total = Total of items extracted on a type 5 edit parameter
Current value = Value of the named work field

Each type of output phase field value is described as follows:

Column Total

A field is assigned a column total if the field is a numeric input or work field that is specified on a type 5 edit parameter. A reference to the field on a type 6 edit parameter or type 8 process parameter obtains the sum of all detail occurrences of the field for the current control break. This is true even for numeric fields that specify format codes FD (date), FS (social security number), and FM (edit mask).

Fields in this category cannot be specified as result fields on a type 8 process parameter. Fields in this category that appear on a SORT parameter are sorted by the extracted value of the field. The value of the field as a sort key, however, cannot be obtained by a reference on a type 6 edit parameter or type 8 process parameter.

Sort-key Value

A field is assigned a sort-key value if it is one of the following:

- A numeric input or work field that appears on a SORT parameter but does not appear on a type 5 edit parameter
- An alphanumeric input or work field that appears on a SORT parameter

A field in this category that is referenced on a type 6 edit parameter or type 8 process parameter is assigned the value of the sort key associated with the last detail line processed before the current control break. Fields in this category cannot be result fields on a type 8 process parameter.

Work Field

A field is assigned the current value of a type 0 or type 1 work field if it is one of the following:

- A numeric work field that does not appear on a SORT parameter or on a type 5 edit parameter
- An alphanumeric work field that does not appear on a SORT parameter

A field in this category that is referenced on a type 6 edit parameter or type 8 process parameter is assigned the current value of the work field. The current value depends on the type of work field, as follows:

- If the work field is a type 0 parameter, the field value at the start of the output phase is the value at the end of the extract phase.
- If the work field is a type 1 parameter, the field value at the start of the output phase is the value at the beginning of the extract phase.

The subscript of a subscripted work field is interpreted as a pointer to an occurrence in an array that holds the current values of the work field. The subscript must be a numeric literal or a numeric work field that does not appear on a type 5 edit parameter or on a SORT parameter and that does not specify a decimal point.

A reference to a field is invalid on a type 6 or type 8 parameter if the field is one of the following:

- A numeric input field that does not appear on a SORT parameter or on a type 5 edit parameter
- An alphanumeric input field that does not appear on a SORT parameter

Examples

The following examples illustrate how CA Culprit interprets references to subscripted fields during the output phase.

Example 1

```
010 AMT.5

0151*005 AMT.1
0151*010 AMT.2
0161*005 AMT.1
0161*010 AMT.2
```

Each occurrence of AMT.1 and AMT.2 is extracted and stored in the extracted items and statistics file during the extract phase. A reference to AMT.1 on a type 6 edit parameter returns the sum of all values extracted by the type 5 edit parameter that references AMT.1.

Example 2

```
010 AMT.5  SUBS

0151*005 AMT.SUBS
0161*010 AMT.SUBS
0162*005 AMT.3
```

During the extract phase, every occurrence of AMT.SUBS is extracted, as indicated by the current value of SUBS. During the output phase, every extracted value of AMT.SUBS is summed, regardless of the run-time value of SUBS during the extract phase. SUBS is the name of a total field during the output phase and no longer has a unique value as a subscript.

The value of AMT.3 during the output phase is the current value of the third occurrence of the total-time array, provided AMT.3 is not specified on a SORT parameter or on a type 5 edit parameter.

Example 3

```
010  AMT.3
0115 TOTAL1  AMT.1
0115 TOTAL2  AMT.2
0151*005 TOTAL1
0151*010 TOTAL2
0161*005 TOTAL1
0161*010 TOTAL2
```

TOTAL1 and TOTAL2 are implied subscript parameters that assign alternative names to the first and second occurrences of the multiply-occurring work field AMT. During the output phase, a reference to TOTAL1 and TOTAL2 yields the total values for each occurrence shown.

Example 4

```
010  AMT.3  SUBS
0115 TOTAL1  AMT.1
0115 TOTAL2  AMT.2
0151*005 AMT.SUBS
0161*005 TOTAL1
0162*010 TOTAL2
```

All extracted values of AMT are accumulated as a single value that is referenced in this example as AMT.SUBS. AMT.SUBS can be referenced during the output phase on a type 6 edit parameter or type 8 process parameter. TOTAL1 and TOTAL2 are references to the current value of the first and second occurrences of the array that stores values of AMT.

Appendix C: Execution JCL

This section contains the following topics:

Overview

This appendix contains the following information:

- The CA Culprit six processing phases in z/OS, z/VSE, z/VM installations
- A discussion of one-step and five-step CA Culprit execution
- z/OS, z/VSE, and z/VM JCL required to run CA Culprit as a one-step or five-step job

CA Culprit Default File Assignments

CA Culprit jobs execute in six phases. The following CA Culprit Processing Phases table lists the six CA Culprit processing phases and their functions. The CA Culprit System Diagram illustrates these processing phases. Introduction (see page 13) contains a detailed description of each processing phase.

Files that are assigned to each processing phase at a z/OS, z/VSE, z/VM installation appear in the tables under Default File Assignments for z/OS, z/VSE, and z/VM Installations and Default File Assignments for z/VSE Installations. These files are used during each processing phase as follows:

Precompile Phase

The precompile (CULP0) phase generates two output files. SYS004 (z/OS, z/VSE, z/VM) or the SYSLST system file contains the Sequential Parameter Listing. SYS005 contains unsorted user input parameters and statistics records that are used to determine resource requirements in subsequent processing phases.

Parameter Sort Phase

The parameter sort phase sorts all run parameters according to CA Culprit's standard sort. These are stored in SYS005.

Note: (z/VM only) To interface CA Culprit with external sort packages, see the *CA IDMS Installation Guide* for your operating system guide.

Compile Phase

The compile (CULL) phase generates a series of machine-language subroutines for each report. If errors are detected during this phase, appropriate compilation diagnostics are output on the Input Parameter Listing, which is stored in SYS004 at z/OS, z/VM, and z/VSE installations.

Extract Phase

The extract (CULL) phase reads the input data file or files, starting with SYS010, and the sorted parameter file from SYS005. In a database access run, the extract phase reads a key file from SYS002.

This phase writes information that pertains to totaling and output fields to the extracted items and statistics file. Items for reports that require sorting are written to file SYS006; items for reports that do not require sorting are written to file SYS008. The extract phase also generates and prints run-time error messages.

Data Sort Phase

The data sort phase sorts record items contained in the extracted items and statistics file (SYS006) and returns the sorted items to the same file.

Note: (z/VM only) To interface CA Culprit with external sort packages, see the CA IDMS installation guide for your operating system guide.

Output Phase

The output (CULE) phase reads the sorted extracted items and statistics file (SYS006) and the unsorted extracted items file (SYS008) for those reports that do not require sorting. This phase outputs printed reports to SYS004; reports written to tape or disk to SYS020, SYS021, and so on; reports written to CARD to SYSPCH; and reports written to special forms to SYS030, SYS031, and so on under z/OS, z/VM and to SYS004 under z/VSE.

The default file names listed in the tables under Default File Assignments for z/OS, z/VSE, and z/VM Installations and Default File Assignments for z/VSE Installations may not be appropriate for some installations. In these cases, the default file assignments can be changed by altering information stored on the installation's PROFILE CSECT. The installation default can also be altered at runtime (for that run only) with five-step JCL for file SYS010 in the CULP2 step and for SYS004 in the CULP4 step.

Note: For more information about PROFILE CSECT, see the *CA IDMS Installation Guide* for your operating system.

Under z/VSE, users can make use of a standard label track for label and extent information on CA Culprit files. In these cases, the installation's PROFILE CSECT may be altered. For example, a standard label track can be established for the file usually assigned to SYS005 by coding the following JCL statements:

```
// DLBL     WORKPARM,'PARMS',0
// EXTENT SYS036,123456,,,190,19
```

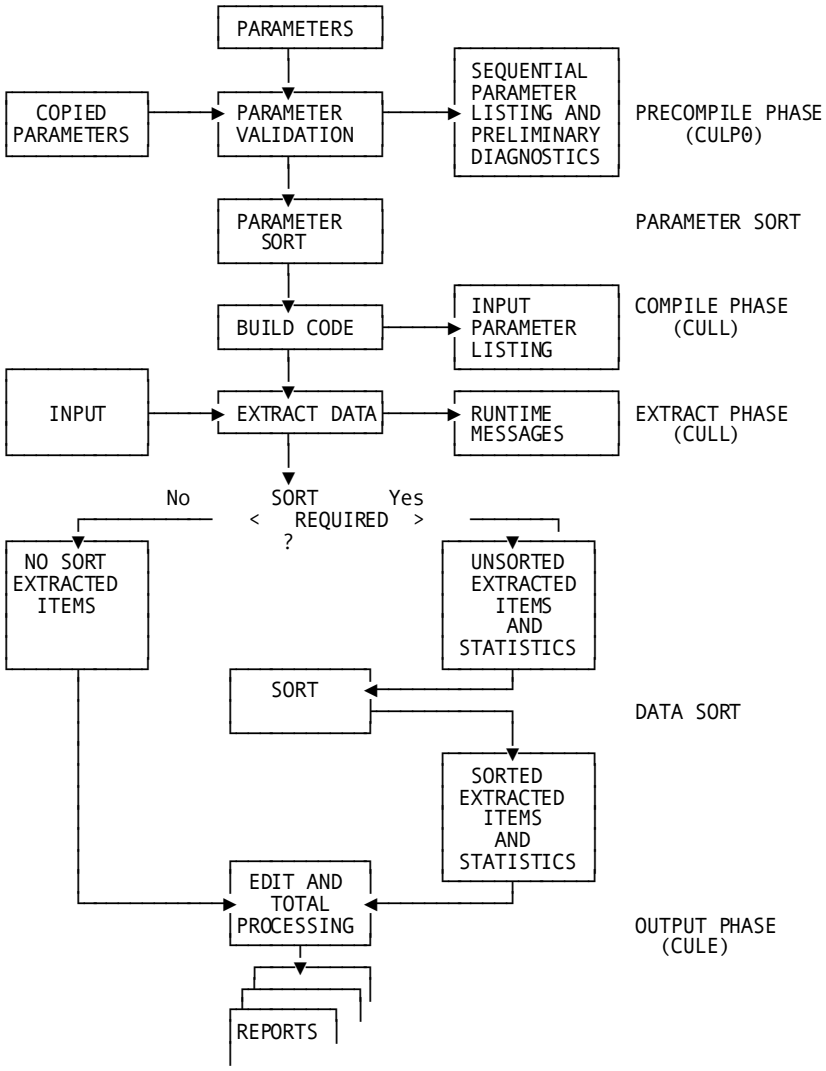
The line that refers to file SYS005 in the PROFILE CSECT must be changed to reflect the new file name and logical unit number.

CA Culprit Processing Phases

Processing Phases* Processing	Function I/O
Precompile (CULP0)	Performs preprocessing functions on CA Culprit parameters
Parameter sort	Sorts the CA Culprit parameters
Compile (CULL)	Generates machine-language subroutines for each report
Extract (CULL)	Reads the input file or files and writes totaling code and edit-control information to the extracted items and statistics file
Data sort	Sorts all items on the extracted items and statistics file
Output (CULE)	Performs automatic totaling and produces printed reports and output files

* Alternative names for each processing phase are included in parentheses ().

CA Culprit System Diagram



Default File Assignments for z/OS, z/VSE, and z/VM Installations

Phase	Mode	Linkname	
Precompile	Output	SYS004 or SYSLST system file	Sequential Parameter Listing; preprocessor diagnostics
	Output	SYS005*	Unsorted, reformatted user parameters and statistics records
Parameter sort	Input	SYS005	Sorted, reformatted user parameters and statistics records

Phase	Mode	Linkname	
	Output	SYS005	Sorted, reformatted user parameters and statistics records
Compile/ Extract	Input	SYS002	Key file for database access runs
	Output	SYS004 or SYSLST system file	Input Parameter Listing; compilation diagnostics; runtime error messages (from extract phase)
	Input	SYS005	Sorted, reformatted user parameters and statistics records
	Output	SYS006*	Unsorted control records and report items for those reports requiring sorting
	Output	SYS007	Generated sort-control statements
	Output	SYS008	Report items for unsorted reports
	Input	SYS010, SYS011, etc.	User input file(s)
Data Sort	Input	SYS006	Sorted control records and report items
	Output	SYS006	Sorted control records and report items
	Input	SYS007	Generated sort-control statements
Output	Output	SYS004	Printed reports; error messages
	Input	SYS006	Sorted control records and report items
	Input	SYS008	Report items for unsorted reports
	Output	SYS020, SYS021, etc.	Nonprinted output for file types PS and IS (each output file is assigned sequentially, starting with SYS020)
	Output	SYS030, SYS031, and so on	Special forms reports (each report is assigned to a sequential output file, starting with SYS030)
	Output	SYSPCH	Card output

* z/VM users must use an external sort package to sort these files.

Default File Assignments for VSE Installations

Processing Phase	I/O Mode	File- Name	Logical Unit	Description
Precompile	Output		SYS004	Sequential Parameter Listing; preprocessor diagnostics
	Output	SYS005	SYS005	Unsorted, reformatted user parameters and statistics records
Parameter sort	Input	SYS005	SYS005	Sorted, reformatted user parameters and statistics records
	Output	SYS005	SYS005	Sorted, reformatted user parameters and statistics records
Compile/ Extract	Input	SYS002	SYS002	Key file for database access runs
	Output		SYS004	Input Parameter Listing; compilation diagnostics; runtime error messages (from extract phase)
	Input	SYS005	SYS005	Sorted, reformatted user parameters and statistics records
	Output	SYS006	SYS006	Unsorted control records and report items for those reports requiring sorting
	Output	SYS007	SYS007	Generated sort-control statements
	Output	SYS008	SYS008	Report items for unsorted reports
	Input	SYS010, SYS011, etc.	SYS010, SYS011, etc.	User input file(s)
Data sort	Input	SYS006	SYS006	Sorted control records and report items
	Output	SYS006	SYS006	Sorted control records and report items
	Input	SYS007	SYS007	Generated sort-control statements

Processing Phase	I/O Mode	File- Name	Logical Unit	Description
Output	Output		SYS004	Printed reports; error messages
	Output	SYS004		Special forms reports (file type NS)
	Input	SYS006	SYS006	Sorted control records and report items
	Input	SYS008	SYS008	Report items for unsorted reports
	Output	SYS020, SYS021, etc.		Nonprinted output for file types PS and IS (each output file is assigned sequentially, starting with SYS020)
	Output	SYSPCH		Card output

One-Step and Five-Step JCL

CA Culprit can execute in one step or five steps at z/OS, z/VSE, and z/VM installations; CA Culprit executes in five steps. When one-step CA Culprit JCL executes, all six processing phases are invoked by a single driver program. When five-step CA Culprit JCL executes, each processing phase executes as a separate step. (The compile and extract phases combine to form one job step.)

The major advantage of one-step CA Culprit is that it eliminates much of the JCL necessary for running the job, which in turn simplifies the system. The single driver program maintains control over all processing phases. The driver controls each of the six processing phases in turn and regains control after each phase except the compile phase.

Before proceeding with the next processing phase, the driver checks the previous phase for a completion code of zero (that is, a normal return). If the completion code is not zero, an error message is printed and the entire job is terminated without executing the remaining processing phases.

A return code for the entire job is set, depending on the reason for termination. The error message associated with job cancellation prints on a new page in the output listing under the heading CA Culprit CONTROLLER MESSAGES. This error message indicates the processing phase that caused job termination and the completion code set by this processing phase.

Note: For more information on error messages, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

With five-step CA Culprit jobs run under z/OS, z/VM, users can control run terminations according to the severity levels of the errors encountered. The control is established through the RC= keyword expression on the PROFILE parameter (see Chapter 2, PROFILE Parameter) and in the execution JCL. At z/VSE installations, the RC= option causes cancellation of the entire job when the indicated error level is encountered.

Differences in Parameter Sort and Data Sort Phases

One-step and five-step CA Culprit jobs invoke the parameter sort and data sort phases differently, as follows:

- Under five-step CA Culprit, both phases are invoked by the JCL.
- Under one-step CA Culprit, both phases are internal to the CA Culprit program. Sort user-exit routines handle the input and output for the sort phases, making SORTIN and SORTOUT DD statements unnecessary. The input and output file for the parameter sort phase is the formatted parameter file, SYS005. The input and output file for the data sort phase is the extracted items and statistics file, SYS006.

  The user can change the name of the sort program invoked by one-step CA Culprit by assembling the system profile option program, CULPPROF, with the SORT= macro operand at installation time. The sort-control statements can also be changed easily, as needed.

  Parameters for CA Culprit's two sort phases are stored as follows:

  - At z/OS installations, parameters for the parameter sort reside in the SRCLIB member, SORT1; parameters for the data sort are generated from the template found in the CULLSRTC module.
  - At z/VSE installations, parameters for the parameter sort reside in the SORT1.SAMPS member in the IDMSBASE sublibrary; parameters for the data sort are generated from the template found in the CULLSRTO relocatable module.
  - At z/VM installations, parameters for the parameter sort reside in IDMSLIB MACLIB member SORT1; parameters for the data sort are generated from the template found in the CULLSRTC module.

Note: For more information, see the *CA IDMS Installation Guide* for your operating system.

Sort listings generated by one-step CA Culprit differ from those generated by five-step CA Culprit. Order of the sort listings is determined by the placement of the sort output DD statements relative to SYS004. The sort listings in one-step CA Culprit either precede or follow the Sequential Parameter and Input Parameter Listings but cannot appear between them. In five-step CA Culprit, the parameter sort listing appears after the Sequential Parameter Listing, and the data sort listing appears after the Input Parameter Listing.

In one-step CA Culprit, the numbers of records processed by each sort are listed under INSERT and DELETE headers; in five-step CA Culprit, the numbers are listed under IN and OUT.

One-Step z/OS JCL— Central Version

Here is sample one-step z/OS JCL to execute CA Culprit batch jobs, when running central version:

CA Culprit One-Step (Central Version) (z/OS)

```
//CULPRIT  EXEC PGM=CULPRIT,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//SORTPRNT DD   SYSOUT=A
//SORTMSG  DD   SYSOUT=A
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS005   DD   DSN=&&sprmwork,DISP=(NEW,DELETE),

//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//SYS006   DD   DSN=&.&sextwork.,DISP=(NEW,DELETE),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYS007   DD   DSN=&&srtpwork.,DISP=(NEW,DELETE),
//             UNIT=disk,SPACE=(TRK,(1,1)),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(NEW,DELETE),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=VB,LRECL=512,BLKSIZE=4628)
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//CULSRT1I DD   DSN=yourHLQ.CAGJSRC(SORT1),DISP=SHR
//SYSIN4   DD   DUMMY,DCB=BLKSIZE=80

//VSAMCTRL DD   DUMMY
//CULLIB   DD   DSN=yourHLQ.CAGJSRC,DISP=SHR
//SYSPCH   DD   SYSOUT=B,DCB=BLKSIZE=80
//SYS002   DD   DSN=user.keyfile,DISP=SHR
//SYS010   DD   DSN=user.inputfil,DISP=OLD,
//             UNIT=tape,VOL=SER=vvvvvv
//SYS011   DD   DSN=user.matchfil,DISP=OLD,
//             UNIT=tape,VOL=SER=vvvvvv
//SYS020   DD   DSN=user.nonprint,DISP=(NEW,CATLG),
//             UNIT=tape,VOL=SER=vvvvvv,
//             DCB=(DSORG=PS,LRECL=llll,BLKSIZE=bbbb)
```

```
//SYS030   DD   SYSOUT=(s,,form),DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIN    DD   *
Put CA Culprit parameter statements here
/*
//*
```

For the description of the previous variables see the following table in One-Step z/OS JCL - Local Mode.

One-Step z/OS JCL—Local Mode

Here is sample one-step z/OS JCL to execute CA Culprit batch jobs in local mode:

CA Culprit one-step (local mode) (z/OS)

```
//CULPRIT  EXEC PGM=CULPRIT,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SYSOUT   DD   SYSOUT=A
//SYSPRINT DD   SYSOUT=A
//SORTPRNT DD   SYSOUT=A
//SORTMSG  DD   SYSOUT=A
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)

//SYS005   DD   DSN=&&sprmwork,DISP=(NEW,DELETE),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//SYS006   DD   DSN=&&sextwork.,DISP=(NEW,DELETE),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYS007   DD   DSN=&&srtpwork.,DISP=(NEW,DELETE),
//              UNIT=disk,SPACE=(TRK,(1,1)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(NEW,DELETE),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=VB,LRECL=512,BLKSIZE=4628)
```

```
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//CULSRT1I DD   DSN=yourHLQ.CAGJSRC(SORT1),DISP=SHR
//SYSIN4   DD   DUMMY,DCB=BLKSIZE=80
//VSAMCTRL DD   DUMMY
//CULLIB   DD   DSN=yourHLQ.CAGJSRC,DISP=SHR
//SYSPCH   DD   SYSOUT=B,DCB=BLKSIZE=80
//SYS002   DD   DSN=user.keyfile,DISP=SHR
//SYS010   DD   DSN=user.inputfil,DISP=OLD,
//              UNIT=tape,VOL=SER=vvvvvv

//SYS011   DD   DSN=user.matchfil,DISP=OLD,
//              UNIT=tape,VOL=SER=vvvvvv
//SYS020   DD   DSN=user.nonprint,DISP=(NEW,CATLG),
//              UNIT=tape,VOL=SER=vvvvvv,
//              DCB=(DSORG=PS,LRECL=llll,BLKSIZE=bbbb)
//SYS030   DD   SYSOUT=(s,,form),DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//dictdb   DD   DSN=idms.appldict.ddldml,DISP=SHR
//dloddb   DD   DSN=idms.appldict.ddldclod,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//userdb   DD   DSN=user.userdb,DISP=SHR
//sysjrnl  DD   DUMMY
//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIN    DD   *
Put CA Culprit parameter statements here
/*
//*
```

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the CA IDMS executable modules
sys1.sortlib	Data set name of system sort library
&&sprmwork	Data set name of the sorted parameter file that is used as the input and output for the parameter sort
disk	Symbolic device name for work files

&&sextwork.	Data set name of extracted items work file, which is used as input and output for data sort; the record length for extracted records must be at least 256 bytes (larger for nonprint records); the block size is typically 1/4 to 1/2 track
&&srtpwork.	Data set name of sort control parameter file
&&nsrtwork.	Data set name of unsorted parameter file output from extract phase; input to output phase
yourHLQ.CAGJSRC(SORT1)	Data set name and member name of stored sort parameters, as established during installation
SYSIN4	Optional restart parameter (DUMMY indicates there are no restart parameters; if SYSIN4 records follow, replace DUMMY with an asterisk)
VSAMCTRL	Used for the optional control cards read by CULLVSAM when that user input module is invoked
yourHLQ.CAGJSRC	Data set name of PDS containing parameters to be copied (required only if USE, =COPY, or =MACRO is used)
SYSPCH	Punched card output
user.keyfile	Data set name for user keyfile
user.inputfil	Data set name for primary input file
tape	Symbolic device name for tape file
vvvvvv	Volume serial number
user.matchfil	Data set name for match file (required only if match file is input; default ddnames for subsequent files are SYS012, SYS013, and so on)
user.nonprint	Data set name for nonprint/nonpunch output (omitted if there are no such files; default ddnames for subsequent files are SYS021, SYS022, and so on)
llll	Logical record length
bbbb	Block size
(s,,form)	SYSOUT class (s) and special forms identification (form) for special forms output (omitted if there are no such files; default ddnames for subsequent special forms output files are SYS031, SYS032, and so on)
dictdb	DDname of the application dictionary definition area
idms.appldict.ddldml	Data set name of the application dictionary definition (DDLDML) area

dloddb	DDname of the application dictionary definition load area
idms.appldict.ddldclod	Data set name of the application dictionary definition load (DDLDCLOD) area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
userdb	DDname of the user database file
user.userdb	Data set name of the user database file
sysjrnl	DDname of the tape journal file
dmcl-name	Specifies the name of the DMCL load module
sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

Database Access

The following JCL modifications must be made in order to access SQL and ASF tables in local mode:

Note: If the user's DMCL load module is not in the CA IDMS/DB load library, then the load library that contains these modules must also be included in the STEPLIB concatenation when running in local mode.

SQL Tables

```
//sqldd    DD   DSN=idms.syssql.ddlcat,DISP=SHR
//sqlxdd   DD   DSN=idms.syssql.ddlcatx,DISP=SHR
```

sqldd	DDname of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	data set name of the SQL catalog (DDLCAT) area
sqlxdd	DDname of the SQL catalog index (DDLCATX) area
idms.syssql.ddlcatx	data set name of the SQL catalog index (DDLCATX) area

ASF Tables

```
//asfdml   DD   DSN=idms.asfdict.ddldml,DISP=SHR
//asflod   DD   DSN=idms.asfdict.asflod,DISP=SHR
//asfdefn  DD   DSN=idms.asfdict.asfdefn,DISP=SHR
//asfdata  DD   DSN=idms.asfdict.asfdata,DISP=SHR
```

asfdml	DDname of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	data set name of the asf dictionary definition (DDLDML) area
asflod	DDname of the asf dictionary definition load (DDLDCLOD) area
idms.asfdict.ddldclod	data set name of the asf dictionary definition load (DDLDCLOD) area
asfdefn	DDname of the asf data definition (IDMSR-AREA) area
idms.asfdict.asfdefn	data set name of the asf data definition area (IDMSR-AREA) area
asfdata	DDname of the asf data (IDMSR-AREA2) area
idms.asfdict.asfdata	data set name of the asf data area (IDMSR-AREA2) area

Five-Step z/OS JCL—Central Version

Here is sample five-step z/OS JCL to execute CA Culprit batch jobs, when running central version:

CA Culprit five-step (central version) (z/OS)

```
//CULP0    EXEC PGM=CULP0,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS005   DD   DSN=&&uprmwork.,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//CULLIB   DD   DSN=yourHLQ.CAGJSRC,DISP=SHR
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSIDMS  DD   *
```

```
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIN    DD   *
Put CA Culprit parameter statements here
/*
//*
//CULP1    EXEC PGM=SORT,REGION=1024K,PARM='MSG=AP'
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTIN   DD   DSN=&&uprmwork.,DISP=(OLD,DELETE)
//SORTOUT  DD   DSN=&&sprmwork,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//SYSOUT   DD   SYSOUT=A
//SYSIN    DD   DSN=yourHLQ.CAGJSRC(SORT1),DISP=SHR
//*

//CULP2    EXEC PGM=CULL,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS005   DD   DSN=&&sprmwork,DISP=(OLD,DELETE)
//SYS006   DD   DSN=&&uextwork.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYS007   DD   DSN=&&srtpwork.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(TRK,(1,1)),
//             DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=VB,LRECL=512,BLKSIZE=4628)
//VSAMCTRL DD   DUMMY
```

```
//SYS002   DD   DSN=user.keyfile,DISP=SHR
//SYS010   DD   DSN=user.inputfil,DISP=OLD,
//             UNIT=tape,VOL=SER=vvvvvv
//SYS011   DD   DSN=user.matchfil,DISP=OLD,
//             UNIT=tape,VOL=SER=vvvvvv
//sysctl   DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR

//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//*
//CULP3    EXEC PGM=SORT,REGION=1024K,PARM='MSG=AP'
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))

//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTIN   DD   DSN=&&uextwork.,DISP=(OLD,DELETE)
//SORTOUT  DD   DSN=&&sextwork.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYSOUT   DD   SYSOUT=A
//SYSIN    DD   DSN=&&srtpwork.,DISP=(OLD,DELETE)
//*
//CULP4    EXEC PGM=CULE,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR

//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS006   DD   DSN=&&sextwork.,DISP=(OLD,DELETE)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(OLD,DELETE)
//SYSPCH   DD   SYSOUT=B,DCB=BLKSIZE=80
//SYS020   DD   DSN=user.nonprint,DISP=(NEW,CATLG),
//             UNIT=tape,VOL=SER=vvvvvv,
//             DCB=(DSORG=PS,LRECL=llll,BLKSIZE=bbbb)
//SYS030   DD   SYSOUT=(s,,form),DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSIN4   DD   DUMMY,DCB=BLKSIZE=80
//*
```

For the description of the previous variables, see the table in Five-Step z/OS JCL—Local mode (see page 331).

Five-Step z/OS JCL—Local mode

Here is sample five-step z/OS JCL to execute CA Culprit batch jobs in local mode:

CA Culprit five-step (local mode) (z/OS)

```
//CULP0    EXEC PGM=CULP0,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS005   DD   DSN=&&uprmwork.,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//CULLIB   DD   DSN=yourHLQ.CAGJSRC,DISP=SHR
//dictdb   DD   DSN=idms.appldict.ddldml,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD   DUMMY
//SYSIDMS  DD   *

DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//SYSIN    DD   *
Put CA Culprit parameter statements here
/*
//*
//CULP1    EXEC PGM=SORT,REGION=1024K,PARM='MSG=AP'
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTIN   DD   DSN=&&uprmwork.,DISP=(OLD,DELETE)
//SORTOUT  DD   DSN=&&sprmwork,DISP=(NEW,PASS),
//             UNIT=disk,SPACE=(CYL,(5,2)),
//             DCB=(RECFM=FB,LRECL=320,BLKSIZE=1600)
//SYSOUT   DD   SYSOUT=A
//SYSIN    DD   DSN=yourHLQ.CAGJSRC(SORT1),DISP=SHR
//*
```

```
//CULP2    EXEC PGM=CULL,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS005   DD   DSN=&&sprmwork,DISP=(OLD,DELETE)
//SYS006   DD   DSN=&&uextwork.,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYS007   DD   DSN=&&srtpwork.,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(TRK,(1,1)),
//              DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=VB,LRECL=512,BLKSIZE=4628)

//VSAMCTRL DD   DUMMY
//SYS002   DD   DSN=user.keyfile,DISP=SHR

//SYS010   DD   DSN=user.inputfil,DISP=OLD,
//              UNIT=tape,VOL=SER=vvvvvv
//SYS011   DD   DSN=user.matchfil,DISP=OLD,
//              UNIT=tape,VOL=SER=vvvvvv
//dloddb   DD   DSN=idms.appldict.ddldclod,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//userdb   DD   DSN=user.userdb,DISP=SHR
//sysjrnl  DD   DUMMY
//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
//*

//CULP3    EXEC PGM=SORT,REGION=1024K,PARM='MSG=AP'
//SORTLIB  DD   DSN=sys1.sortlib,DISP=SHR
//SORTWK01 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK02 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK03 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTWK04 DD   UNIT=disk,SPACE=(CYL,(5,2))
//SORTIN   DD   DSN=&&uextwork.,DISP=(OLD,DELETE)
//SORTOUT  DD   DSN=&&sextwork.,DISP=(NEW,PASS),
//              UNIT=disk,SPACE=(CYL,(5,2)),
//              DCB=(RECFM=VB,LRECL=2044,BLKSIZE=4628)
//SYSOUT   DD   SYSOUT=A
//SYSIN    DD   DSN=&&srtpwork.,DISP=(OLD,DELETE)
//*
```

```
//CULP4    EXEC PGM=CULE,REGION=1024K
//STEPLIB  DD   DSN=idms.dba.loadlib,DISP=SHR
//         DD   DSN=idms.loadlib,DISP=SHR
//SYS004   DD   SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS006   DD   DSN=&&sextwork.,DISP=(OLD,DELETE)
//SYS008   DD   DSN=&&nsrtwork.,DISP=(OLD,DELETE)
//SYSPCH   DD   SYSOUT=B,DCB=BLKSIZE=80
//SYS020   DD   DSN=user.nonprint,DISP=(NEW,CATLG),
//              UNIT=tape,VOL=SER=vvvvvv,
//              DCB=(DSORG=PS,LRECL=llll,BLKSIZE=bbbb)
//SYS030   DD   SYSOUT=(s,,form),DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYSIN4   DD   DUMMY,DCB=BLKSIZE=80
//*
```

idms.dba.loadlib	Data set name of the load library containing the DMCL and database name table load modules
idms.loadlib	Data set name of the load library containing the CA IDMS executable modules
&&uprmwork.	Data set name of unsorted parameter file that contains the input for the first sort step
disk	Symbolic device name for work files
yourHLQ.CAGJSRC	Data set name of PDS containing parameters to be copied (required only if USE, =COPY, or =MACRO is used)
dictdb	DDname of the application dictionary definition area
idms.appldict.ddldml	Data set name of the application dictionary definition (DDLDML) area
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysjrnl	DDname of the tape journal file
dmcl-name	Specifies the name of the DMCL load module
sys1.sortlib	Data set name of system sort library
&&sprmwork	Data set name of the sorted parameter file that contains the output from the first sort step
yourHLQ.CAGJSRC(SORT1)	Data set name and member name of stored sort parameters, as established during installation

&&uextwork.	Data set name of extracted items work file, which is used as input for data sort; the record length for extracted records must be at least 256 (larger for nonprint records); the block size is typically 1/4 to 1/2 track
&&srtpwork.	Data set name of sort control file
&&nsrtwork.	Data set name of unsorted parameter file output from extract phase; input to output phase
VSAMCTRL	Used for the optional control cards read by CULLVSAM when that user input module is invoked
user.keyfile	Data set name for user keyfile
user.inputfil	Data set name for primary input file
tape	Symbolic device name for tape file
vvvvvv	Volume number
user.matchfil	Data set name for match file (required only if match file is input; default ddnames for subsequent files are SYS012, SYS013, and so on)
dloddb	DDname of the application dictionary definition load area
idms.appldict.ddldclod	Data set name of the application dictionary definition load (DDLDCLOD) area
userdb	DDname of the user database file
user.userdb	Data set name of the user database file
&&sextwork.	Data set name of sorted extracted items file
SYSPCH	Punched card output
user.nonprint	Data set name for nonprint/nonpunch output (omitted if there are no such files; default ddnames for subsequent files are SYS021, SYS022, and so on)
llll	Logical record length
bbbb	Block size
(s,,form)	SYSOUT class (s) and special forms identification (form) for special forms output (omitted if there are no such files; default ddnames for subsequent special forms output files are SYS031, SYS032, and so on)
SYSIN4	Optional restart parameter (DUMMY indicates there are no restart parameters; if SYSIN4 records follow, replace DUMMY with an asterisk)

sysctl	DDname of the SYSCTL file
idms.sysctl	Data set name of the SYSCTL file

Database Access

Make the following additions when accessing SQL or ASF tables:

Creating or Updating Tables—Central Version

When creating or updating SQL or ASF tables, while running under central version, also add the following statements to CULP4:

```
//sysctl DD   DSN=idms.sysctl,DISP=SHR
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
```

sysctl	DDname of the SYSCTL file
idms.sysctl	data set name of the SYSCTL file
dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	data set name of the system message (DDLDCMSG) area
dmcl-name	Specifies the name of the DMCL load module

Extracting Data from SQL Tables—Local Mode

When running local mode, add the following DD statements to the CULP2 and CULP4 steps to extract data from an SQL table:

```
//sqldd    DD   DSN=idms.syssql.ddlcat,DISP=SHR
//sqlxdd   DD   DSN=idms.syssql.ddlcatx,DISP=SHR
```

When writing to an SQL table, the following DD statement is also required in the CULP4 step:

```
//userdb   DD   DSN=user.userdb,DISP=SHR
```

sqldd	DDname of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	data set name of the SQL catalog (DDLCAT) area
sqlxdd	DDname of the SQL catalog index (DDLCATX) area
idms.syssql.ddlcatx	data set name of the SQL catalog index (DDLCATX) area

userdb	DDname of the user database file
user.userdb	data set name of the user database file

Note: If the user's DMCL load module is not in the CA IDMS/DB load library, then the load library that contains these modules must also be included in the STEPLIB concatenation when running in local mode.

Extracting Data from ASF Tables—Local Mode

When running local mode, add the following DD statements to the CULP2 and CULP4 steps to extract data from an ASF table. Use the same DD statements when writing to an ASF table in the CULP4 step:

```
//asfdml   DD   DSN=idms.asfdict.ddldml,DISP=SHR
//asflod   DD   DSN=idms.asfdict.asflod,DISP=SHR
//asfdefn  DD   DSN=idms.asfdict.asfdefn,DISP=SHR
//asfdata  DD   DSN=idms.asfdict.asfdata,DISP=SHR
```

asfdml	DDname of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	data set name of the asf dictionary definition (DDLDML) area
asflod	DDname of the asf dictionary definition load (DDLDCLOD) area
idms.asfdict.ddldclod	data set name of the asf dictionary definition load (DDLDCLOD) area
asfdefn	DDname of the asf data definition (IDMSR-AREA) area
idms.asfdict.asfdefn	data set name of the asf data definition area (IDMSR-AREA) area
asfdata	DDname of the asf data (IDMSR-AREA2) area
idms.asfdict.asfdata	data set name of the asf data area (IDMSR-AREA2) area

Note: If the user's DMCL load module is not in the CA IDMS/DB load library, then the load library that contains these modules must also be included in the STEPLIB concatenation when running in local mode.

Creating or Updating Tables— Local Mode

When creating or updating SQL or ASF tables, while running local mode, also add the following statements to CULP4:

```
//dcmsg    DD   DSN=idms.sysmsg.ddldcmsg,DISP=SHR
//sysjrnl  DD   DUMMY
//SYSIDMS  DD   *
DMCL=dmcl-name
Put other SYSIDMS parameters, as appropriate, here
/*
```

dcmsg	DDname of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	data set name of the system message (DDLDCMSG) area
sysjrnl	DDname of the tape journal file
dmcl-name	Specifies the name of the DMCL load module

One-Step z/VSE JCL—Central Version

Here is sample one-step z/VSE JCL to execute CA Culprit batch jobs, when running central version:

CA Culprit one-step (central version) (z/VSE)

```
// JOB    CULP1STP
// DLBL   idmslib,'idms.library'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA IDMS libraries
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'sprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'sextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS007,'srtpwork',0
```

```
// EXTENT SYS007,vvvvvv,,,ssss,tttt
// ASSGN  SYS007,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSPH,'syspch.dsn'
// EXTENT SYSPCH,vvvvvv,,,ssss,tttt
// ASSGN  SYSPCH,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.keyfile'
// EXTENT SYS002,vvvvvv,,,ssss,tttt

// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// TLBL   SYS010,'user.inputfil'
// ASSGN  SYS010,cuu
// TLBL   SYS011,'user.matchfil'
// ASSGN  SYS011,cuu
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// EXEC   PROC=sysctl
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',,DA
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   Culprit,SIZE=1024K
  Put CA Culprit commands here
/*
  Put user input here if on cards
/*
  Restart parameter
/*
/&
```

One-Step z/VSE JCL—Local Mode

Here is sample one-step z/VSE JCL to execute CA Culprit batch jobs in local mode:

CA Culprit one-step (local mode) (z/VSE)

```
// JOB    CULP1STP
// DLBL   idmslib,'idms.library'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS libraries
// EXEC PROC=idmslbls
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'sprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'sextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt

// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS007,'srtpwork',0
// EXTENT SYS007,vvvvvv,,,ssss,tttt
// ASSGN  SYS007,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSPH,'syspch.dsn'
```

```
// EXTENT SYSPCH,vvvvvv,,,ssss,tttt
// ASSGN  SYSPCH,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.keyfile'
// EXTENT SYS002,vvvvvv,,,ssss,tttt
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// ASSGN  sysxxx,IGN                        *This is for SYSJRNLS*
// TLBL   SYS010,'user.inputfil'
// ASSGN  SYS010,cuu
// TLBL   SYS011,'user.matchfil'
// ASSGN  SYS011,cuu
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   Culprit,SIZE=1024K
  Put CA Culprit commands here
/*
  Put user input here if on cards
/*
  Restart parameter
/*
/&
```

idmslib	File Name of the CA IDMS library
'idms.library'	File-id of the CA IDMS library, as established during installation
sysxxx	SYS number
vvvvvv	Volume serial number
ssss	Starting extent
tttt	Number of tracts
CA IDMS libraries	The CA IDMS libraries, as established during installation
idmslbls	A procedure containing file definitions for dictionaries, sample databases, disk journal files, and the SYSIDMS file. For more information, see IDMSLBLS Procedure (see page 351).
sprmwork	Data set name of sorted parameters file that contains the output from the first sort step
srtpwork	Data set name of the sort-control parameter file contains the output from the first sort step

sextwork	Data set name of sorted extracted items file
nsrtwork	Data set name of unsorted extract output data set
srtwk1.fileid	Data set name of sort workfile
srtwk2.fileid	Data set name of sort workfile
srtwk3.fileid	Data set name of sort workfile
syspch.dsn	File for punched card output
user.keyfile	Data set name of a key file (required only if a key file is input)
user.inputfil	Data set name for primary input file
cuu	Address of the tape device
user.matchfil	Data set name for match file (default ddnames for subsequent files are SYS012, SYS013, and so on)
user.nonprint	Data set name for nonprint/nonpunch output (omitted is there are no such files; default ddnames for subsequent nonprint/nonpunch output files are SYS021, SYS022, on so on)
sysctl	Procedure name that contains the SYSCTL file
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysidms.parms	Data set name of the SYSIDMS file that contains the dmcl-name and other appropriate SYSIDMS parameters
dmcl-name	Name of the DMCL

Five-Step z/VSE JCL—Central Version

Here is sample five-step z/VSE JCL to execute CA Culprit batch jobs, when running central version:

CA Culprit five-step (central version) (z/VSE)

```
// JOB    CULP5STP
// DLBL   idmslib,'idms.library'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS libraries
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'uprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// EXEC   PROC=sysctl
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',,DA
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULP0,SIZE=1024K
  Put CA Culprit commands here
/*

// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK4,'sortwk4.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SYS005,'uprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'sprmwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// EXEC   SORT,SIZE=1024K
* $$ SLI MEM=SORT1.SAMPS,S=idmslib.sublibrary
/*
```

```
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'sprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'uextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS007,'srtpwork',0
// EXTENT SYS007,vvvvvv,,,ssss,tttt
// ASSGN  SYS007,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR

// DLBL   SYS002,'user.keyfile'
// EXTENT SYS002,vvvvvv,,,ssss,tttt
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// TLBL   SYS010,'user.inputfil'
// ASSGN  SYS010,cuu
// TLBL   SYS011,'user.matchfil'
// ASSGN  SYS011,cuu
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// EXEC   PROC=sysctl
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',,DA
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULL,SIZE=1024K
  Put user input here if on cards
/*
```

```
// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK4,'sortwk4.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTIN1,'uextwork',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTOUT,'sextwork',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSIN,'srtpwork'

// EXTENT SYSIPT,vvvvvv,,,ssss,tttt
// ASSGN  SYSIPT,DISK,VOL=vvvvvv,SHR
// EXEC   SORT,SIZE=1024K
/*
// ASSGN  SYS004,SYSLST
// DLBL   SYS006,'sextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSPH,'srtpwork'
// EXTENT SYSPCH,vvvvvv,,,ssss,tttt

// ASSGN  SYSPCH,DISK,VOL=vvvvvv,SHR
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSIGN sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULE,SIZE=1024K
   Restart parameter
/*
/&
```

Five-Step z/VSE JCL—Local Mode

Here is sample five-step z/VSE JCL to execute CA Culprit batch jobs in local mode:

CA Culprit five-step (local mode) (z/VSE)

```
// JOB    CULP5STP
// DLBL   idmslib,'idms.library'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// LIBDEF *,SEARCH=CA-IDMS libraries
// EXEC PROC=idmslbls
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'uprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// ASSGN  sysxxx,IGN          *ASSGN SYSJRNL to IGN*
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULP0,SIZE=1024K
  Put CA Culprit commands here
/*

// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK4,'sortwk4.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SYS005,'uprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'sprmwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// EXEC   SORT,SIZE=1024K
* $$ SLI MEM=SORT1.SAMPS,S=idmslib.sublibrary
/*
```

```
// ASSGN  SYS004,SYSLST
// DLBL   SYS005,'sprmwork',0
// EXTENT SYS005,vvvvvv,,,ssss,tttt
// ASSGN  SYS005,DISK,VOL=vvvvvv,SHR
// DLBL   SYS006,'uextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS007,'srtpwork',0
// EXTENT SYS007,vvvvvv,,,ssss,tttt
// ASSGN  SYS007,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
// DLBL   SYS002,'user.keyfile'
// EXTENT SYS002,vvvvvv,,,ssss,tttt
// ASSGN  SYS002,DISK,VOL=vvvvvv,SHR
// TLBL   SYS010,'user.inputfil'
// ASSGN  SYS010,cuu

// TLBL   SYS011,'user.matchfil'
// ASSGN  SYS011,cuu
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',,DA
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULL,SIZE=1024K
  Put user input here if on cards
/*
// DLBL   SORTWK1,'sortwk1.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK2,'sortwk2.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK3,'sortwk3.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
```

```
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTWK4,'sortwk4.fileid',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTIN1,'uextwork',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   SORTOUT,'sextwork',0
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSIN,'srtpwork'
// EXTENT SYSIPT,vvvvvv,,,ssss,tttt
// ASSGN  SYSIPT,DISK,VOL=vvvvvv,SHR
// EXEC   SORT,SIZE=1024K
/*

// ASSGN  SYS004,SYSLST
// DLBL   SYS006,'sextwork',0
// EXTENT SYS006,vvvvvv,,,ssss,tttt
// ASSGN  SYS006,DISK,VOL=vvvvvv,SHR
// DLBL   SYS008,'nsrtwork',0
// EXTENT SYS008,vvvvvv,,,ssss,tttt
// ASSGN  SYS008,DISK,VOL=vvvvvv,SHR
// DLBL   IJSYSPH,'srtpwork'
// EXTENT SYSPCH,vvvvvv,,,ssss,tttt
// ASSGN  SYSPCH,DISK,VOL=vvvvvv,SHR
// TLBL   SYS020,'user.nonprint'
// ASSGN  SYS020,cuu
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
// EXEC   CULE,SIZE=1024K
   Restart parameter
/*
/&
```

idmslib	Filename of the CA IDMS library
'idms.library'	File-id of the CA IDMS library, as established during installation
sysxxx	SYS number
vvvvvv	Volume serial number
ssss	Starting extent
tttt	Number of tracks
idmslib.sublibrary	Base sub-library name in *idms.library*

CA IDMS libraries	The CA IDMS libraries, as established during installation
idmslbls	A procedure containing file definitions for dictionaries, sample databases, disk journal files, and the SYSIDMS file. For more information, see IDMSLBLS Procedure (see page 351), later in this chapter.
uprmwork	Data set name of unsorted parameter file
sysctl	Procedure name that contains the SYSCTL file
idms.sysmsg.ddldcmsg	Data set name of the system message (DDLDCMSG) area
sysidms.parms	Data set name of the SYSIDMS file that contains the dmcl-name and other appropriate SYSIDMS parameters
dmcl-name	Name of the DMCL
srtwk1.fileid	Data set name of sort workfile
srtwk2.fileid	Data set name of sort workfile
srtwk3.fileid	Data set name of sort workfile
sprmwork	Data set name of sorted parameters file that contains the output from the first sort step
uextwork	Data set name of unsorted extracted item data set; the record length for extracted records must be in the range 256 - 1024 bytes; the block size is typically 1/4 to 1/2 track; CA Culprit copies this DCB information to SYS008
sextwork	Data set name of sorted extracted items file
srtpwork	Data set name of the sort-control parameter file
nsrtwork	Filename of unsorted extract output data set
user.keyfile	Data set name of a key file (required only if a key file is input)
user.inputfil	Data set name for primary input file
cuu	Address of the tape device
user.matchfil	Data set name for match file (default ddnames for subsequent files are SYS012, SYS013, and so on)

user.nonprint	Data set name for nonprint/nonpunch output (omitted is there are no such files; default ddnames for subsequent nonprint/nonpunch output files are SYS021, SYS022, on so on)

Database Access

The following JCL modifications must be made to the CULP0, CULP2, and CULP4 steps in order to access non-SQL defined database records or tables; the run must execute each step under the central version of CA IDMS/DB in order to access tables:

- In the CULP0 step, the following modifications are required:
  - If the CULP0 step is executed under the central version, add the following statement before the //EXEC CULP0 statement:

    ```
    // UPSI     xxxxxxxx       if specified in the IDMSOPTI module
    ```

    Note: To override the IDMSOPTI specification, define a SYSCTL file in the JCL:

    ```
    // DLBL     sysctl,'idms.sysctl',,DA
    // EXTENT   sys017,nnnnnn
    // ASSGN    sys017,DISK,VOL=nnnnnn,SHR
    ```

  - If the CULP0 step is run locally, JCL statements must be included for the user's CA IDMS/DB data dictionary and its associated journal files. The following JCL must be added between the // EXTENT SYS005 and // ASSGN SYSSLB statements:

    ```
    // ASSGN   sys009,IGN
      additional journal-file assignments, as required
    // ASSGN   sys015,X,'ddd'
    // DLBL    dictdb,'idms.dictdb',,DA
    // EXTENT  sys015,X,nnnnnn,,,ssss,llll
    ```

- In the CULP2 step, the following modifications are required:
  - If the CULP2 step is executed under the central version, add the following statement before the //EXEC CULL statement:

    ```
    // UPSI     xxxxxxxx       if specified in the IDMSOPTI module
    ```

    Note: To override the IDMSOPTI specification, define a SYSCTL file in the JCL:

    ```
    // DLBL     sysctl,'idms.sysctl',,DA
    // EXTENT   sys017,nnnnnn
    // ASSGN    sys017,DISK,VOL=nnnnnn,SHR
    ```

- If the CULP2 step is run locally, JCL statements must be included for the user's database and its associated journal files. The following JCL must be added between the // TLBL SYS011 and // EXEC CULL statements:

```
// ASSGN   sys009,IGN
   additional journal-file assignments, as required
// ASSGN   sys018,X,'ddd'
// DLBL    userdb,'user.userdb',,DA
// EXTENT  userdb,X,nnnnnn,,,ssss,llll
  additional database-file assignments, as required
```

- To access a key file, add the following statements after the // ASSGN SYS004 statement:

```
// ASSGN  SYS002,X'ppp'
// TLBL   SYS002,'user.keyfile'
```

- In the CULP4 step, the following statement is required to access tables; add the statement before the // EXEC CULE statement:

```
// UPSI    xxxxxxxx      if specified in the IDMSOPTI module
```

Note: To override the IDMSOPTI specification, define a SYSCTL file in the JCL:

```
// DLBL     sysctl,'idms.sysctl',,DA
// EXTENT   sys017,nnnnnn
// ASSGN    sys017,DISK,VOL=nnnnnn,SHR
```

xxxxxxxx	Appropriate UPSI value (default installation is 1)
idms.dictdb	File-ID of the data dictionary file
idms.sysctl	File-ID of the SYSCTL file
dictdb	Filename of the data dictionary file
nnnnnn	Volume serial number
sysctl	Filename of the SYSCTL file
sys009	Filename for journal file as assigned at CA IDMS/DB installation (ASSGN omitted if input is other than CA IDMS/DB database)
sys015	Filename for data dictionary as assigned at CA IDMS/DB installation (omitted if input is other than CA IDMS/DB database)
sys017	Logical unit assignment of the SYSCTL file
sys018	Filename for CA IDMS/DB database as assigned in the schema definition (required if input is CA IDMS/DB; omitted if it is not)

userdb	Filename of the database file
user.keyfile	File-ID of the key file (required only if a key file is input)
user.userdb	File-ID of the CA IDMS/DB database

IDMSLBLS Procedure

What Is the IDMSLBLS Procedure?

IDMSLBLS is a procedure provided during a CA IDMS z/VSE installation. It contains file definitions for the CA IDMS components listed as follows. These components are provided during installation:

- Dictionaries
- Sample databases
- Disk journal files
- SYSIDMS file

Tailor the IDMSLBLS procedure to reflect the filenames and definitions in use at your site and include this procedure in z/VSE JCL job streams.

The sample z/VSE JCL provided in this document includes the IDMSLBLS procedure. Therefore, individual file definitions for CA IDMS dictionaries, sample databases, disk journal files, and SYSIDMS file are not included in the sample JCL.

IDMSLBLS Procedure Listing

```
/*  ------------------------ LABELS ------------------------
// DLBL    dccat,'idms.system.dccat',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,31
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatl,'idms.system.dccatlod',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dccatx,'idms.system.dccatx',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcdml,'idms.system.ddldml',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclod,'idms.system.ddldclod',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,21
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclog,'idms.system.ddldclog',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
```

```
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcrun,'idms.system.ddldcrun',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,68
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcscr,'idms.system.ddldcscr',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,135
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dcmsg,'idms.sysmsg.ddldcmsg',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dclscr,'idms.sysloc.ddlocscr',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirldb,'idms.sysdirl.ddldml',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dirllod,'idms.sysdirl.ddldclod',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,2

// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empdemo,'idms.empdemo1',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    insdemo,'idms.insdemo1',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    orgdemo,'idms.orgdemo1',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    empldem,'idms.sqldemo.empldemo',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,11
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    infodem,'idms.sqldemo.infodemo',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    projdem,'idms.projseg.projdemo',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    indxdem,'idms.sqldemo.indxdemo',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,6
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sysctl,'idms.sysctl',2099/365,SD
```

```
// EXTENT  SYSnnn,nnnnnn,,,ssss,2
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    secdd,'idms.sysuser.ddlsec',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dictdb,'idms.appldict.ddldml',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    dloddb,'idms.appldict.ddldclod',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqldd,'idms.syssql.ddlcat',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqllod,'idms.syssql.ddlcatl',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,51
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    sqlxdd,'idms.syssql.ddlcatx',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,26
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdml,'idms.asfdict.ddldml',2099/365,DA

// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asflod,'idms.asfdict.asflod',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,401
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    asfdata,'idms.asfdict.asfdata',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,201
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    ASFDEFN,'idms.asfdict.asfdefn',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,101
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j1jrnl,'idms.j1jrnl',2099/365,DA

// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j2jrnl,'idms.j2jrnl',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL    j3jrnl,'idms.j3jrnl',2099/365,DA
// EXTENT  SYSnnn,nnnnnn,,,ssss,54
// ASSGN   SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL   SYSIDMS,'sysidms.parms'
// EXTENT sysxxx,vvvvvv,,,ssss,tttt
// ASSGN  sysxxx,DISK,VOL=vvvvvv,SHR
/+
/*
```

dccat	Filename of the system dictionary catalog (DDLCAT) area
idms.system.dccat	File-ID of the system dictionary catalog (DDLCAT) area
SYS.nnn	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial identifier of appropriate disk volume
ssss	Starting track (CKD) or block (FBA) of disk extent
dccatl	Filename of the system dictionary catalog load (DDLCATLOD) area
idms.system.dccatlod	File-ID of the system dictionary catalog load (DDLCATLOD) area
dccatx	Filename of the system dictionary catalog index (DDLCATX) area
idms.system.dccatx	File-ID of the system dictionary catalog index (DDLCATX) area
dcdml	Filename of the system dictionary definition (DDLDML) area
idms.system.ddldml	File-ID of the system dictionary definition (DDLDML) area
dclod	Filename of the system dictionary definition load (DDLDCLOD) area
idms.system.ddldclod	File-ID of the system dictionary definition load (DDLDCLOD) area
dclog	Filename of the system log area (DDLDCLOG) area
idms.system.ddldclog	File-ID of the system log (DDLDCLOG) area
dcrun	Filename of the system queue (DDLDCRUN) area
idms.system.ddldcrun	File-ID of the system queue (DDLDCRUN) area
dcscr	Filename of the system scratch (DDLDCSCR) area
idms.system.ddldcscr	File-ID of the system scratch (DDLDCSCR) area
dcmsg	Filename of the system message (DDLDCMSG) area
idms.sysmsg.ddldcmsg	File-ID of the system message (DDLDCMSG) area
dclscr	Filename of the local mode system scratch (DDLOCSCR) area
idms.sysloc.ddlocscr	File-ID of the local mode system scratch (DDLOCSCR) area
dirldb	Filename of the IDMSDIRL definition (DDLDML) area
idms.sysdirl.ddldml	File-ID of the IDMSDIRL definition (DDLDML) area

dirllod	Filename of the IDMSDIRL definition load (DDLDCLOD) area
idms.sysdirl.dirllod	File-ID of the IDMSDIRL definition load (DDLDCLOD) area
empdemo	Filename of the EMPDEMO area
idms.empdemo1	File-ID of the EMPDEMO area
insdemo	Filename of the INSDEMO area
idms.insdemo1	File-ID of the INSDEMO area
orgdemo	Filename of the ORGDEMO area
idms.orgdemo1	File-ID of the ORDDEMO area
empldem	Filename of the EMPLDEMO area
idms.sqldemo.empldemo	File-ID of the EMPLDEMO area
infodem	Filename of the INFODEMO area
idms.sqldemo.infodemo	File-ID of the INFODEMO area
projdem	Filename of the PROJDEMO area
idms.projseg.projdemo	File-ID of the PROJDEMO area
indxdem	Filename of the INDXDEMO area
idms.sqldemo.indxdemo	File-ID of the INDXDEMO area
sysctl	Filename of the SYSCTL file
idms.sysctl	File-ID of the SYSCTL file
secdd	Filename of the system user catalog (DDLSEC) area
idms.sysuser.ddlsec	File-ID of the system user catalog (DDLSEC) area
dictdb	Filename of the application dictionary definition area
idms.appldict.ddldml	File-ID of the application dictionary definition (DDLDML) area
dloddb	Filename of the application dictionary definition load area
idms.appldict.ddldclod	File-ID of the application dictionary definition load (DDLDCLOD) area
sqldd	Filename of the SQL catalog (DDLCAT) area
idms.syssql.ddlcat	File-ID of the SQL catalog (DDLCAT) area
sqllod	Filename of the SQL catalog load (DDLCATL) area
idms.syssql.ddlcatl	File-ID of SQL catalog load (DDLCATL) area
sqlxdd	Filename of the SQL catalog index (DDLCATX) area

idms.syssql.ddlcatx	File-ID of the SQL catalog index (DDLCATX) area
asfdml	Filename of the asf dictionary definition (DDLDML) area
idms.asfdict.ddldml	File-ID of the asf dictionary definition (DDLDML) area
asflod	Filename of the asf dictionary definition load (ASFLOD) area
idms.asfdict.asflod	File-ID of the asf dictionary definition load (ASFLOD) area
asfdata	Filename of the asf data (ASFDATA) area
idms.asfdict.asfdata	File-ID of the asf data area (ASFDATA) area
ASFDEFN	Filename of the asf data definition (ASFDEFN) area
idms.asfdict.asfdefn	File-ID of the asf data definition area (ASFDEFN) area
j1jrnl	Filename of the first disk journal file
idms.j1jrnl	File-ID of the first disk journal file
j2jrnl	Filename of the second disk journal file
idms.j2jrnl	File-ID of the second disk journal file
j3jrnl	Filename of the third disk journal file
idms.j3jrnl	File-ID of the third disk journal file
SYSIDMS	Filename of the SYSIDMS parameter file that contains the dmcl-name and other appropriate SYSIDMS parameters

One-Step z/VM Commands—Central Version

Here are sample one-step z/VM commands to execute CA Culprit batch jobs, when running central version:

CA Culprit one-step (central version) (z/VM)

```
SET STORECLR ENDCMD
FILEDEF SYSIN DISK culprit code a
FILEDEF SYSIDMS DISK sysidms parms a
FILEDEF SYS020 DISK user nonprt a
FILEDEF SYS011 DISK user match a
FILEDEF SYS010 DISK user input a
FILEDEF SYS030 DISK forms id a
FILEDEF SORTLIB
EXEC CULPFD
EXECOS OSRUN CULPRIT PARM='optional execution parameters'
```

sysidms parms a	Filename, filetype, and filemode of the file containing SYSIDMS parameters for the Culprit run
culprit code a	Filename, filetype, and filemode of the file containing the Culprit code
forms id a	Filename, filetype, and filemode of the file containing the special forms identification, if any
optional execution parameters	Optional parameters to control the Culprit run
user input a	Filename for the primary input file
user match a	Filename for the match file (required only when the match file is input; subsequent files, by default, reference SYS012, SYS013, and so on).
user nonprt a	Filename for the nonprint/nonpunch output (omitted when there are no such files; subsequent nonprint/nonpunch files reference SYS021, SYS022, and so on)

Usage

SYSIDMS File

To run Culprit, you should include these SYSIDMS parameters:

- DMCL=*dmcl-name*, to identify the DMCL
- If you are running Culprit against an SQL-defined database, DBNAME=*dictionary-name*, to identify the dictionary whose catalog component contains the database definitions

To create the SYSIDMS file of SYSIDMS parameters:

1. On the VM/ESA command line, type:

   ```
   XEDIT sysidms parm a (NOPROF
   ```

2. Press [Enter]
3. On the XEDIT command line, type:

   ```
   INPUT
   ```

4. Press [Enter]
5. In input mode, type in the SYSIDMS parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:

   FILE
8. Press [Enter]

Note: For more information on documentation of SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

One-Step z/VM Commands—Local Mode

To specify that Culprit is executing in local mode, perform one of the following:

- Link Culprit with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter of the filename, type and mode identified by *sysidms parm a*.
- Modify the OSRUN statement:

  ```
  OSRUN Culprit PARM='*LOCAL*'
  ```

  Note: This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.

Database Access

The following modifications must be made in order to access non-SQL defined database records or tables:

- To run CA Culprit locally, add the following commands for the CA IDMS/DB data dictionary and user database:

  ```
  FILEDEF dictdb DISK dictdb_addr
  FILEDEF userdb DISK userdb_addr
  ```

- To access a key file, add the following command:

  ```
  FILEDEF SYS002 DISK keyfile FILE A3
  ```

Note: CA Culprit, CA IDMS/DB users who wish to override IDMSOPTI specifications at runtime can code either the LOCAL keyword or CVMACH= keyword expression on the DATABASE parameter, discussed in Chapter 11, DATABASE Parameter. For more information about IDMSOPTI, see the *CA IDMS System Operations Guide*.

dictdb	DDname of the data dictionary DDLDML area
dictdb.addr	Minidisk address of the database file (e.g., 500)

keyfile	Filename of the key file (required only if a key file is input)
userdb	Filename of the CA IDMS/DB database, as assigned in the schema definition
userdb_addr	Minidisk address of the database file (e.g., 500)

Five-Step z/VM Commands—Central Version

Here are sample five-step VM/ESA commands to execute CA Culprit batch jobs, when running central version:

CA Culprit five-step (central version) (VM/ESA)

```
*----------------------CULP0---------------------------------*
FILEDEF sysctl DISK sysctl idms a
FILEDEF dcmsg DISK idms dmsgdb
FILEDEF SYSIDMS DISK sysidms parms a
FILEDEF SYSIPT DISK sysipt input a
FILEDEF SYS005 DISK unsorted parms a (LRECL 320 BLKSIZE 1600 RECF FB
EXEC CULPFD
EXECOS OSRUN CULP0
*----------------------CULP1---------------------------------*
SORT CULSYS05 PARAM A CULSRT05 PARAM A CULSORT1 SYSIN A
FILEDEF SYS005 DISK CULSRT05 PARAM A (RECFM FB LRECL 320 BLKSIZE 1600

FILEDEF SYS006 DISK CULSYS06 EXTRACT A (RECFM VB LRECL 1024 BLKSIZE 1028
FILEDEF SYS007 DISK CULSYS07 SORTCARD A (RECFM F LRECL 80 BLKSIZE 80
FILEDEF SYS008 DISK CULSYS08 NOSORT A (RECFM VB LRECL 1024 BLKSIZE 1028
FILEDEF SYS010 DISK culprit cards a (RECFM F LRECL 80
FILEDEF SYS002 DISK keyfile data a (RECFM F LRECL 80
FILEDEF SYS011
*----------------------CULP2---------------------------------*
EXECOS OSRUN CULL
*----------------------CULP3---------------------------------*
SORT CULSYS06 EXTRACT A CULSRT06 EXTRACT A CULSYS07 SORTCARD A
FILEDEF SYS020 DISK SPFLOAD DATA A (LRECL 80 BLKSIZE 3200 RECFM FB
*----------------------CULP4---------------------------------*
EXECOS OSRUN CULE
```

sysctl	DDname of the SYSCTL file
sysctl idms a	Filename of the SYSCTL file
culprit cards a	Culprit card input file
idms sysctl a	Filename of the file containing the SYSCTL file parameters

dcmsg	DDname of the system message (DDLDCMSG) area
idms dmsgdb	Minidisk address of the message database (e.g., 500) (DDLDCMSG) area
ppp	Page size of the database file
nnn	Number of pages in the database file
sysidms parms a	Filename, filetype, and filemode of the file containing SYSIDMS parameters for the Culprit run
sysipt input a	Filename of the file containing the Culprit input parameters
unsorted parms a	Filename of unsorted parameter file that contains the input for the first sort step
CULPFD	Exec which defines all FILEDEFs, TXTLIBs, and LOADLIBs required by the system
keyfile data a	Filename of a key file (required only if a key file is input)

Usage

SYSIDMS File

To create the SYSIDMS file of SYSIDMS parameters:

1. On the VM/ESA command line, type:

   ```
   XEDIT sysidms parms a (NOPROF
   ```

2. Press [Enter]
3. On the XEDIT command line, type:

   ```
   INPUT
   ```

4. Press [Enter]
5. In input mode, type in the SYSIDMS parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:

   ```
   FILE
   ```

8. Press [Enter]

Note: For more information on documentation of SYSIDMS parameters, see the *CA IDMS Common Facilities Guide*.

SYSIPT File

To create the SYSIPT file of Culprit input parameters:

1. On the VM/ESA command line, type:

   ```
   XEDIT sysipt data a (NOPROF
   ```

2. Press [Enter]
3. On the XEDIT command line, type:

   ```
   INPUT
   ```

4. Press [Enter]
5. In input mode, type in the Culprit input parameters
6. Press [Enter] to exit input mode
7. On the XEDIT command line, type:

   ```
   FILE
   ```

8. Press [Enter]

SYSIDMS File

To run Culprit, you should include these SYSIDMS parameters:

- DMCL=*dmcl-name*, to identify the DMCL
- If you are running Culprit against an SQL-defined database, DBNAME=*dictionary-name*, to identify the dictionary whose catalog component contains the database definitions

Five-Step z/VM Commands—Local Mode

To specify that Culprit is executing in local mode, perform one of the following:

- Link Culprit with an IDMSOPTI program that specifies local execution mode
- Specify *LOCAL* as the first input parameter of the filename, type and mode identified by *sysidms parm a* in the CULPFD exec.
- Modify the OSRUN statement:

  ```
  OSRUN Culprit PARM='*LOCAL*'
  ```

  Note: This option is valid only if the OSRUN command is issued from a System Product interpreter or an EXEC2 file.

Database Access

The following JCL modifications must be made in order to access non-SQL defined database records and tables:

- If the CULP0 step is to be run locally, add the following commands for the CA IDMS/DB data dictionary and associated journal files:

```
FILEDEF dictdb DISK dictdb dictfile
FILEDEF j1jrnl DUMMY
FILEDEF j2jrnl DUMMY
          additional journal-file commands, as required
```

- If the CULP2 step is to be run locally, add the following commands for the user's database and associated journal file:

```
FILEDEF userdb DISK user userdb
          additional database-file commands, as required
```

- If a key file is to be used, add the following command in the CULP2 step:

```
FILEDEF SYS002 DISK keyfile FILE A3
```

Note: CA Culprit, CA IDMS/DB users who wish to override IDMSOPTI specifications at runtime can code either the LOCAL keyword or CVMACH= keyword expression on the DATABASE parameter, discussed in Chapter 11, DATABASE Parameter. For more information about IDMSOPTI, see *CA IDMS System Operations Guide*.

dictdb	DDname of the data dictionary DDLDML area
dictdb dictfile	Minidisk address of the database file (e.g., 500)
j1jrnl	DDname of the first disk journal file
j2jrnl	DDname of the second disk journal file
keyfile	Filename of the key file (required only if a key file is input)
userdb	DDname of the user database file
user userdb	Minidisk address of the user database file (e.g., 500)

Appendix D: Restart Capability

This section contains the following topics:

Overview

A restart capability is built into the CA Culprit output phase to permit independent execution of the output phase for any or all of the reports produced by the run. This capability allows the following changes from the typical run procedures:

- One run can produce reports on different types of output paper, such as special forms. To print reports on different types of paper, the output phase is run separately for each report or group of reports that uses the same type of output medium.
- More than one copy of a report can be printed by running the output phase two or more times.
- Execution of the output phase can be resumed after an interruption occurs during a print operation.

Using Restart

To use the restart capability, a RESTART parameter must be defined in the CA Culprit execution JCL.

More information:

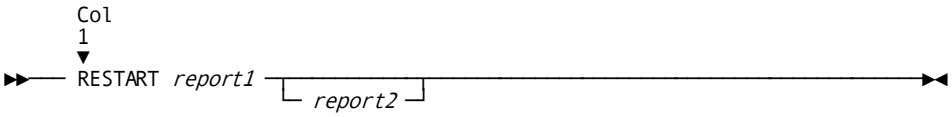
Execution JCL (see page 315)

RESTART

Purpose

Permits independent execution of the output phase for any or all of the reports produced by the run.

Syntax



Parameters

RESTART

Identifies the parameter type. Must be coded starting in column 1.

report1

Identifies the first report to be printed by this execution of the output phase. *Report1* must be a number in the range 00 through 99 and must be separated from RESTART by at least one space.

report2

Identifies the last report to be printed by this execution of the output phase. *Report2* must be a number in the range 00 through 99 and must be equal to or greater than *report1*. If a value is specified, it must follow *report1* with no intervening spaces.

Usage

The following considerations apply to coding *report2*:

- If *report2* is not specified, the default is EOF; that is, all reports from *report1* to the last report in the run are printed.
- If *report2* equals *report1*, one report is printed.
- If *report2* is greater than *report1*, all reports from *report1* to *report2* are printed.

Note: (z/OS users) Printing a range of reports or one report can be handled more efficiently by altering CA Culprit execution JCL.

Appendix E: Release 5 and 6 Default Actions

This section contains the following topics:

What is RELEASE=5?

Compatibility with Release 5 and Earlier

Culprit Releases 6.0 and later provide a feature that enables users to run programs written under earlier Culprit releases without modification. At installation time, the default processing mode is set to either RELEASE=5, which runs programs using default actions available with Culprit releases prior to 6.0, or RELEASE=6, which runs programs using default actions available with Culprit Releases 6.0 and later.

When to Use It

If a site has a large number of programs written under Culprit releases earlier than 6.0, RELEASE=5 should be specified at installation. Otherwise, RELEASE=6 is recommended. At run time, the installation processing mode can be overridden by using the RELEASE=5/6 keyword expression on the PROFILE parameter. All features of Releases 6.0 and later are available, regardless of the release option specified. The release option affects only certain default actions taken by Culprit.

More information:

PROFILE Parameter (see page 29)

Default Actions

Internal Formats

Some of the default actions taken by Culprit in the two processing modes are based on the internal format that Culprit uses to store numeric fields. Under Culprit Release 5.0, all user-defined numeric input and work fields and extracted-item and total-bucket values for these fields were stored as 8-byte packed values. Under Culprit Release 6.0 and later, these items are stored as 8-byte or 16-byte packed values, regardless of the release option specified, as follows:

- The following fields and their associated extracted-item and total-bucket values are stored as **16-byte** fields:
  - **Numeric input fields** that specify a DP= keyword expression or that, due to size and data type, contain more than 15 digits
  - **Numeric work fields** that specify a DP= keyword expression, an initial value with a decimal point, or an initial value that contains more than 15 digits
- All other numeric input and work fields and their associated extract-item and total-bucket values are stored as **8-byte** packed values.

The following table summarizes default actions taken by Culprit under RELEASE=5 and RELEASE=6. These differences are discussed more fully below.

Culprit Facility	Release 5 Default action	Release 6 Default action
Number of decimal positions printed for numeric output fields	2	0
Treatment of arithmetic and MOVE operation results	TRUNCATE	ROUND
Size of intermediate work fields	8-byte packed decimal unless user-defined fields used in the calculation are stored as more than 8-byte packed decimal	16-byte packed decimal
Decimal arithmetic overflows	Overflows truncate the high-order digits; no error message is printed	Overflows invoke the extended error-handling facility

Culprit Facility	Release 5 Default action	Release 6 Default action
Report headers	Headers print anywhere on a report page if a title parameter is not specified	Headers always print either at the top of a new page, if a title parameter is not specified, or under the title if a title parameter is specified

Default actions taken by Culprit under RELEASE=5 and RELEASE=6 are explained below.

Decimal Positions

The number of decimal positions in output fields are handled in the following manner:

- Under **RELEASE=5**, two decimal positions are printed for all numeric fields unless a DP= keyword expression is specified on a REC, work field, or edit parameter for the field.
- Under **RELEASE=6**, no decimal positions are printed unless a DP= keyword expression is specified on a REC, work field, or edit parameter for the field.

Under both RELEASE=5 and RELEASE=6, a DP= keyword expression is ignored for output fields that specify format codes FP, FZ, FM, FD, FS, FB, FU, and FW.

Truncation/Rounding

Numeric results generated in Culprit procedure logic are truncated or rounded, as follows:

- Under **RELEASE=5**, the results of either a MOVE operation or simple or compound arithmetic operations are truncated, except on COMPUTE statements that specify ROUND.
- Under **RELEASE=6**, the results of either a MOVE operation or simple or compound arithmetic operations are rounded, except on COMPUTE statements that specify TRUNCATE.

Work Field Storage

Internal work fields for compound arithmetic operations (that is, arithmetic operations that specify COMPUTE) are handled as follows:

- Under **RELEASE=5**, intermediate work fields are stored as 8-byte packed fields, provided that all user-defined fields in the COMPUTE statements are stored as 8-byte fields. If any are stored as 16-byte fields, all intermediate work fields are stored as 16-byte packed fields, regardless of the processing mode in effect.
- Under **RELEASE=6**, intermediate work fields are stored as 16-byte packed fields.

Note: A COMPUTE instruction that includes multiplication by a power of 10 to shift high-order digits from an intermediate work field will produce different results depending on the processing mode in effect.

Decimal Overflow

Decimal arithmetic overflows in computed results are handled as follows:

- Under **RELEASE=5**, decimal arithmetic overflows that occur in a multiplication operation do not produce an error message; the overflow causes high-order digits to be truncated without warning.
- Under **RELEASE=6**, a decimal arithmetic overflow results when an attempt is made to store a computed result that contains more than 15 significant digits in an 8-byte packed work field. The overflow invokes the extended error-handling facility.

  Note: For more information on the extended error-handling facility, see the *CA Culprit for CA IDMS Messages and Codes Guide*.

Report Headers

Report headers are handled as follows:

- Under **RELEASE=5**, report headers print anywhere on a page unless a title is specified and a space code of 1 (that is, a space code that designates a new page) is indicated for the first header line.
- Under **RELEASE=6**, report headers print at the top of a new page unless a title is specified. If a title is specified, the title prints at the top of the page and headers follow the title, as determined by user-coded space codes.

Appendix F: Reserved Words

This section contains the following topics:

About Reserved Words

What They Are

Reserved words are words that have a special meaning to CA Culprit. Reserved words appear in parameter coding instructions throughout this manual. Some of the reserved words pertain to reports generated from a CA IDMS/DB database.

When to Avoid Them

CA Culprit reserved words should not appear as user-coded field names on REC parameters or work field definition parameters.

Reserved Word Listing

A listing of CA Culprit reserved words follows:

A	GOTO	PICK
AND	HEAD	REC-NAME
ARG*n*	IDMS-STATUS	RELS
B	IF	RETURN
BUFFER	KEY-VALUE	ROUND
C	KEYFILE	S*ID
CALL	LEVL	STOP
COMPUTE	LR-STATUS	STOP-RPT
CONVERT	M	STOP-RUN
D	M*ID	SUBSCHEMA-NAME
DATE	MOVE	TABLE-ID
DB-EXIT	NOSORT	TABLE-NAME
DBKEY	NUMERIC	TAKE
DBKEY-ALPHA	O*ID	TO
DBKEY-LINE	OR	TRUNCATE(TRUNC)
DB-PAGE	PAGE	UNPICK
DROP	PATH-ID	US*nn*
EOF	PERFORM	

Appendix G: Storing and Accessing CA Culprit Code

This section contains the following topics:

Overview

Parameters copied by copied code parameters must be stored in a library prior to run time and must be made available to the precompile phase at execution. This appendix includes the information necessary to store the code, and the z/OS, z/VSE, and z/VM JCL statements necessary to access the stored code.

The stored parameters must be in card-image format and must be copied from one of the following sources: library library

- A z/OS partitioned data set (PDS)
- A z/VSE LIBR dataset
- A CA Panvalet library
- A CA Librarian library
- A z/VM MACLIB
- CA IDMS/DB Integrated Data Dictionary (IDD)

Libraries Searched

The library that is searched for the named member is specified on the PROFILE parameter with the PARMLIB= keyword expression. The PROFILE parameter must appear before the first USE parameter, =COPY statement, or =MACRO statement in order to access the library. Otherwise, the parameters are copied from the library selected as the default for PARMLIB at installation.

Note: For details on establishing installation defaults, see the CA IDMS installation guide for your operating system.

Storing and accessing parameters under z/OS, z/VSE, and z/VM, is discussed separately below.

More information:

Copied Code Parameters (see page 149)
PROFILE Parameter (see page 29)

Storing and Acessing Parameters Under z/OS

To store parameters as a member of a PDS, use a utility program such as IEBGENER. The JCL statements follow:

IEBGENER (z/OS)

```
//     EXEC  PGM=IEBGENER
//SYSPRINT  DD  SYSOUT=A
//SYSUT1    DD  DSN=sequential.data.set,DISP=(OLD,KEEP)
//SYSUT2    DD  DSN=culprit.srclib,DISP=(OLD,KEEP)
//SYSIN     DD  *
   GENERATE MAXNAME=1
   MEMBER NAME=member-name
/*
//
```

culprit.srclib	PDS into which CA Culprit parameters are to be copied and stored
member-name	name under which parameters are to be copied in the PDS
sequential.data.set	name of the sequential data set containing CA Culprit parameters to be copied

At run time, submit the name of the PDS in the //CULLIB DD job control parameter, as shown in Appendix C, Execution JCL. CA Culprit will search the named library for the member specified on the USE parameter, =COPY parameter, or =MACRO parameter.

IDD

To store parameters as modules in the CA IDMS/DB Integrated Data Dictionary (IDD), see the *CA IDMS IDD DDDL Reference Guide*. At run time, submit the name of the data dictionary and its associated journal file(s) in the appropriate DD job control statements, as shown in Appendix C, Execution JCL. CA Culprit retrieves the module specified on the USE parameter, =COPY parameter, or =MACRO parameter from the named data dictionary.

Storing and Accessing Parameters Under z/VSE

Storage as Bookname

To store parameters as a member of a LIBR library, use the following JCL statements:

LIBR (z/VSE)

```
//  EXEC LIBR
  ACCESS SUBLIB=lib.sublib
  CATALOG member.A
  culprit parameters
/+
/*
```

lib.sublib	LIBR library and sublibrary to which you want to add the member
lib.member	Name of the member to be added

IDD

To store parameters as modules in IDD, see the *CA IDMS IDD DDDL Reference Guide*. At run time, submit JCL statements for the data dictionary and its associated journal file(s), as shown in Appendix C, Execution JCL. CA Culprit retrieves the module specified on the USE parameter, =COPY parameter, or =MACRO parameter from the named data dictionary.

Storing and Accessing Parameters Under z/VM

To store parameters as a member of a z/VM MACLIB, follow these steps:

1. Create a file that contains the CA Culprit parameters. The file type of the file must be COPY or MACRO. The file must contain fixed-length 80-byte records.
2. Issue the following command to **create** a MACLIB and store your CA Culprit code:

   `MACLIB GEN filename`
3. Issue the following command to **add** a member to an existing MACLIB:

   `MACLIB ADD filename`

Appendix H: Sample Programs—Conventional File Access

This section contains the following topics:

About This Appendix

This appendix contains three sample CA Culprit programs that read conventional files, CA Culprit parameters coded for the report, and the report output.

Example 1—Employee Compensation Status Report

This program reads a file that contains employee data and produces a report that lists the employees in alphabetical order, the employee's salary status, and the employee's salary.

The first out of the two following figures shows the contents of the input file for the CA Culprit run. The second one is the Sequential Parameter Listing for the CA Culprit run; this listing shows the CA Culprit parameters coded for Report 01. The third figure illustrates the output for Report 01.

Parameters coded for the report are described below:

Input Definition Parameters

The input file contains fixed-length 80-byte records. The input fields required by the CA Culprit run are EMP-NAME, EMP-LNAME, and SALARY. Each field is defined on a REC parameter. The first two fields are alphanumeric; EMP-LNAME redefines bytes 11 through 25 of EMP-NAME. SALARY is a 10-byte packed decimal field with two implied decimal places. An auto-header is specified for EMP-NAME.

Output Definition Parameters

The output for Report 01 is a details-only printed report with 70 characters per line. The detail lines sort in alphabetical order by employee last name. Report 01 contains a title line, column headers, and detail lines; it does not contain total lines because the OUTPUT parameter specifies a details-only report.

The third figure shows a title line printed at the top of the report page. The blank line that follows the title line is specified by header line 1 on the type 4 edit parameter. Header line 2 specifies column headers for relative column numbers 10, 20, and 30. Two blank lines follow the column headers because a spacing code appears in column 10 of the type 4 edit parameter that specifies the literal 'SALARY'.

Report 01 contains three detail lines. Detail line 1 contains the employee's name and salary status. Detail line 2 contains the employee's annual salary if the employee is salaried; detail line 3 contains the employee's hourly wages if the employee is paid on an hourly basis. Detail lines 2 and 3 specify the same output field size, format code, and spacing code; the spacing code overprints on the output line.

Work Field Parameters

Report 01 contains two report-specific work fields. MESSAGE is an alphanumeric work field initialized to eight blanks. Different literal values can be moved to MESSAGE by using process parameter statements.

HR-SALARY is a numeric work field that contains two implied decimal places. The output field size for HR-SALARY is 31 digits because a decimal place is specified. The SZ= keyword expression on the type 5 edit parameter overrides the default output field size for this field.

Process Parameters

The type 7 procedure logic for Report 01 determines whether the employee earns an annual salary of $19,000 or less. If the test is true, the procedure logic branches to sequence number 500 where the employee's hourly salary is computed, the literal 'HOURLY' is moved to the work field MESSAGE, and detail lines 1 and 3 are selected for extracting. If the test is not true, the literal 'SALARIED' is moved to MESSAGE, and detail lines 1 and 2 are selected for extracting.

Input File Layout for the Employee Compensation Status Report:

```
            Record Size:     80 bytes
            Block Size:      80 bytes
            Record Format:   Fixed
Data Field      Start Position   Length (bytes)   Data Type
Employee name     1               25             Alphanumeric
Department name   30              20             Alphanumeric
Job title         50              20             Alphanumeric
Salary            70              10             Packed decimal
```

CA Culprit Parameters for the Employee Compensation Status Report:

```
mm/dd/yy                    SEQUENTIAL PARAMETER LISTING            Vnn.n  PAGE    1
00 ** SYSIN **                          IN  80
  INSTALLATION SECURITY OPTION IS NO
                                        REC EMP-NAME    1 25         'EMPLOYEE' 'NAME'
                                        REC EMP-LNAME  11 15
                                        REC SALARY     70 10 3 DP=2
                                        01OUT  70 D
                                        01SORT EMP-LNAME
                                        010MESSAGE '        '  HR-SALARY DP=2
                                        013EMPLOYEE COMPENSATION STATUS
                                        01410001 ' '
                                        0142*010 'EMPLOYEE'
                                        0142*020 'STATUS'
                                        0142*0300'SALARY'
                                        0151*010 EMP-NAME
                                        0151*020 MESSAGE
                                        0152*030+SALARY      SZ=8 F$
                                        0153*030+HR-SALARY   SZ=8 F$
                                        017010   IF SALARY LE 19000 500
                                        017      MOVE 'SALARIED' TO MESSAGE
                                        017      TAKE (1 2)
                                        017500   MOVE 'HOURLY  ' TO MESSAGE
                                        017      COMPUTE (SALARY DIVIDE 52) DIVIDE 40  HR-SALARY
                                        017      TAKE (1 3)
```

Employee Compensation Status Output:

```
REPORT NO. 01       EMPLOYEE COMPENSATION STATUS mm/dd/yy PAGE     1

              EMPLOYEE              STATUS          SALARY


      ROY       ANDALE              SALARIED       $33,500.00
      MICHAEL   ANGELO              HOURLY             $8.65
      HARRY     ARM                 SALARIED       $46,000.00
      MONTE     BANK                SALARIED       $80,000.00
      JUNE      BLOOMER             HOURLY             $7.21
      CHARLES   BOWER               SALARIED       $38,000.00
      C.        BREEZE              SALARIED       $38,000.00
      TERRY     CLOTH               SALARIED       $38,000.00
      BETH      CLOUD               SALARIED       $52,750.00
      HERBERT   CRANE               SALARIED       $75,000.00
      CAROLYN   CROW                SALARIED       $37,500.00
      ALAN      DONOVAN             SALARIED       $33,500.00
      JANE      DOUGH               SALARIED       $33,000.00
      JANE      FERNDALE            SALARIED       $22,500.00
      PHINEAS   FINN                SALARIED       $45,000.00
      TOM       FITZHUGH            HOURLY             $6.25
      GEORGE    FONRAD              HOURLY             $7.09
      JAMES     GALLWAY             SALARIED       $33,000.00
      ROBIN     GARDNER             HOURLY             $6.73
      JENNIFER  GARFIELD            SALARIED       $65,000.00
      PERCY     GRANGER             SALARIED       $34,500.00
      VLADIMIR  HEAROWITZ           SALARIED       $33,000.00
      HENRIETTA HENDON              SALARIED      $240,000.00
      EDWARD    HUTTON              SALARIED       $44,000.00
      JOCK      JACKSON             SALARIED       $34,000.00
      JAMES     JACOBI              SALARIED       $55,000.00
      JULIE     JENSEN              SALARIED       $37,000.00
      RUPERT    JENSON              SALARIED       $82,000.00
      CYNTHIA   JOHNSON             HOURLY             $6.49
      DOUGLAS   KAHALLY             SALARIED       $20,000.00
      JOE       KASPAR              SALARIED       $31,000.00
      MARIANNE  KIMBALL             SALARIED       $45,000.00
      DORIS     KING                HOURLY             $6.97
      TERENCE   KLWELLEN           SALARIED       $43,000.00
      SANDY     KRAAMER             HOURLY             $6.73
      BURT      LANCHESTER          SALARIED       $54,500.00
      HERBERT   LIPSICH             HOURLY             $8.89
      LARRY     LITERATA            SALARIED       $37,500.00
      RENE      MAKER               SALARIED       $85,000.00
      CAROL     MCDOUGALL           HOURLY             $8.65
      DANIEL    MOON                SALARIED       $72,000.00
      RICHARD   MUNYON              SALARIED       $36,000.00
      BRIAN     NICEMAN             HOURLY             $6.73
      KATHERINE O'HEARN             SALARIED       $42,500.00
      MADELINE  ORGRATZI            SALARIED       $39,000.00
      THEMIS    PAPAZEUS            SALARIED      $100,000.00
      LAURA     PENMAN              SALARIED       $39,000.00
      ELEANOR   PEOPLES             SALARIED       $80,000.00
      JOHN      RUPEE               SALARIED       $80,000.00
      NANCY     TERNER              HOURLY             $6.25
      MARK      TIME                SALARIED       $33,000.00
      RALPH     TYRO                SALARIED       $20,000.00
      RICHARD   WAGNER              SALARIED       $47,000.00
      ROGER     WILCO               SALARIED       $80,000.00
```

Example 2—Average Salaries by Job Title Report

This program reads an input file containing employee salary and job information. The output lists the employees that share the same job title, the salary of each employee, and the average salary for each type of job. The program performs the following functions:

- Sorts input records by employee last name for each job title
- Executes a control break each time the job title changes and at the end of the output phase
- Totals employee salaries at every control break
- Counts the number of employees at every control break
- Calculates an average salary at every control break
- Prints title, header, detail, and total lines

CA Culprit parameters used in this report are shown in the table under Sequential Parameter Listing for Average Salaries by Job Title. The report output is illustrated in the table under Output for Average Salaries by Job Title.

Parameters coded for this report are described below:

Input Definition Parameters

The Input File Layout for the Employee Compensation Status Report shows the input file for this CA Culprit run. The input file contains fixed-length 80-byte records; by default, the DD statement labeled SYS010 in z/OS execution JCL defines the input file. The CA Culprit run uses four input fields; all of the fields are alphanumeric except the SALARY field, which is defined as a 10-byte packed decimal value with two implied decimal positions.

Output Definition Parameters

Report 01 is a printed report with 70 characters per line and 55 lines per page. The detail lines sort in alphabetical order by the last name of each employee that shares the same job title. The control break code on the SORT parameter specifies two blank output lines each time the job title changes.

The type 5 edit parameters coded for this report define one detail line. The detail line contains three printed output fields in relative column positions 10, 20, and 30, and a nonprinted output field that acts as counter. An output field size and format edit code is specified for SALARY.

The type 6 edit parameters coded for this report specify two total lines. The first total line specifies a literal work field value in relative column 10 and the total salary for the current control break in relative column 30. The accumulated value of SALARY prints because SALARY is a numeric field that is referenced on a type 5 edit parameter. A blank line prints between the last detail line and the first total line at each control break because this edit line specifies a spacing code in column 10.

The second total line specifies a literal work field value and the average salary for the current control break. CA Culprit calculates the value of AVG-SAL in type 8 procedure logic.

Work Field Parameters

Report 01 defines two numeric work fields and one multiply-occurring alphanumeric work field, as follows:

- COUNT is a numeric work field that is initialized to 1. It is referenced both as a nonprinted numeric work field on a type 5 edit parameter and as an operand in an arithmetic process operation in type 8 procedure logic.
- LABEL is a multiply-occurring alphanumeric work field that occurs two times. The initial value of each occurrence is a 20-blank literal. Explicit occurrences of LABEL are referenced on type 6 edit parameters and in MOVE operations on type 8 procedure logic.
- AVG-SAL is a numeric work field that is initialized to 0.00. It is referenced as a result field in an arithmetic calculation and is output on a type 6 edit parameter.

Process Parameters

Type 8 procedure logic executes at every control break. The procedure logic for this report determines whether the number of employee records for the current control break exceeds 1; that is, whether the value for COUNT is greater than 1. If the value of COUNT is 1, processing for the control break stops and no total lines print. If the value of COUNT is greater than 1, CA Culprit calculates the average salary for the current control break.

If the level of the control break is 1 (that is, the control break occurs on a title name), the procedure logic branches to sequence number 600. The logic starting at sequence number 600 instructs CA Culprit to move literals to the first and second occurrences of LABEL. An implicit TAKE after sequence number 650 outputs both total lines.

In this example, a control break level not equal to 1 implies that the control break level is 2; a level-2 control break generates the grand total lines for Report 01. At the end of the output phase, CA Culprit moves 'COMPANY SALARY' to LABEL.1. The procedure logic branches to sequence number 650, where CA Culprit moves a literal value to LABEL.2 and outputs the grand total lines for the report.

Sequential Parameter Listing for Average Salaries by Job Title:

```
mm/dd/yy                SEQUENTIAL PARAMETER LISTING                  Vnn.n  PAGE   1
00 ** SYSIN **                          IN 80
  INSTALLATION SECURITY OPTION IS NO
                                        REC EMP-NAME       1   25          'EMPLOYEE' 'NAME'
                                        REC EMP-LNAME     11   10
                                        REC DEPARTMENT    30   20
                                        REC TITLE         50   20
                                        REC SALARY        70   10   3  DP=2
                                        010UT  75
                                        01SORT TITLE,-,EMP-LNAME
                                        013AVERAGE SALARIES BY JOB TITLE
                                        01410001  ' '
                                        0151*000  COUNT
                                        0151*010  TITLE
                                        0151*020  EMP-NAME
                                        0151*030  SALARY  SZ=10 F$
                                        0161*0100 LABEL.1
                                        0161*030  SALARY  SZ=10 F$
                                        0162*010  LABEL.2
                                        0162*030  AVG-SAL SZ=10 F$
                                        010 COUNT 1  LABEL.2 '                    ' AVG-SAL DP=2
                                        018    COUNT EQ 1 DROP
                                        018    SALARY DIVIDE COUNT AVG-SAL
                                        018    LEVL EQ 1 600
                                        018    MOVE 'COMPANY SALARY      ' TO LABEL.1
                                        018    B 650
                                        018600 MOVE 'TOTAL SALARY        ' TO LABEL.1
                                        018650 MOVE 'AVERAGE SALARY      ' TO LABEL.2
```

Output for Average Salaries by Job Title:

```
REPORT NO. 01        AVERAGE SALARIES BY JOB TITLE    mm/dd/yy PAGE     3

   PROGRAMMER/ANALYST      JANE      DOUGH                $33,000.00
   PROGRAMMER/ANALYST      JAMES     GALLWAY              $33,000.00
   PROGRAMMER/ANALYST      PERCY     GRANGER              $34,500.00
   PROGRAMMER/ANALYST      VLADIMIR  HEAROWITZ            $33,000.00
   PROGRAMMER/ANALYST      JULIE     JENSEN               $37,000.00
   PROGRAMMER/ANALYST      KATHERINE O'HEARN              $42,500.00

   TOTAL SALARY                                         $213,000.00
   AVERAGE SALARY                                        $35,500.00

   RAINDANCE CONSULTANT    CAROLYN   CROW                 $37,500.00

   RAINMAKER               BURT      LANCHESTER           $54,500.00

   RECRUITER/INTERVWR      MADELINE  ORGRATZI             $39,000.00

   SNOWBLOWER              ROY       ANDALE               $33,500.00
   SNOWBLOWER              RICHARD   MUNYON               $36,000.00

   TOTAL SALARY                                          $69,500.00
   AVERAGE SALARY                                        $34,750.00

   SPORTS CONSULTANT       JOCK      JACKSON              $34,000.00

   STURM/DRANG ADMIN       HARRY     ARM                  $46,000.00
   STURM/DRANG ADMIN       RICHARD   WAGNER               $47,000.00

   TOTAL SALARY                                          $93,000.00
   AVERAGE SALARY                                        $46,500.00

   SUNSHINE SUPERVISOR     ALAN      DONOVAN              $33,500.00

   SYSTEMS PROGRAMMER      TERENCE   KLWELLEN             $43,000.00

   WINTERIZER              JOE       KASPAR               $31,000.00
   WINTERIZER              MARK      TIME                 $33,000.00

   TOTAL SALARY                                          $64,000.00
   AVERAGE SALARY                                        $32,000.00

   COMPANY SALARY                                     $2,522,500.00
   AVERAGE SALARY                                        $45,044.64
  RECORDS WRITTEN FOR REPORT 01—       118
```

The last page of output is shown.

Example 3— Employee Earning Potential

The input file to this program contains the job title, dates of employment, and salary level for past and present employees of the company. The input file also contains salary levels that correspond to each job title.

The following figure shows the contents of the input file for the CA Culprit run. The input file contains a field that occurs four times; this multiply-occurring field contains the salary levels assigned to each job title.

```
Record Size:        80 bytes
Block Size:       3600 bytes
Record Format:    Fixed

Data Field        Start Position    Length     Data Type
Employee ID           1               4        Zoned decimal
Starting date         6               6        Alphanumeric
Finish date          12               6        Alphanumeric
Salary level         18               2        Zoned decimal
Title                20              20        Alphanumeric
Number of positions  40               3        Zoned decimal
Available positions  43               3        Zoned decimal
Salary levels for    46               2        Zoned decimal
 each job title                       (four times)
```

Input File Layout for the Employee Earning Potential Report:

This program performs the following functions:

- Sorts the detail lines by job title.
- Determines whether an employee is currently employed or was once employed with the specified job title.
- Compares the employee's salary to possible salary levels. If the salary is not in error or at the highest level for the job, the program outputs future salary levels available to the employee.

The parameters coded for this report appear in H7 and are described below. The report output appears in H8.

Input Definition Parameters

Library member JOBRECS stores the input parameters for this report. CA Culprit copies these parameters into the parameter stream during the precompile phase. The run uses six of the nine input fields contained in JOBRECS. START-DATE, FINISH-DATE, and TITLE define alphanumeric fields; EMP-ID, SALARY, and the multiply-occurring field GRADE define zoned decimal fields.

Output Definition Parameters

Report 01 is a printed report that contains title lines, header lines, and detail lines; total lines are suppressed by the details-only specification on the OUTPUT parameter. Type 4 edit parameters specify literals in absolute column positions on three header lines; the third header line specifies a single blank, which generates a blank line between the header and detail lines for Report 01.

Report 01 specifies two detail lines. The first detail line contains the employee's identification number, job title, and salary. CA Culprit prints the identification number with leading zeros because the edit parameter specifies format code FN.

The second detail line contains the employee's start date, termination date (if any), a message that indicates the employee's salary status, and a work field indicating future salary levels.

The type 5 edit parameter that specifies EMP-ID also specifies a spacing code in column 10. CA Culprit outputs a blank line before each detail line 1 is printed.

Work Field Parameters

Report 01 specifies five work fields, as follows:

- INDEX defines a numeric work field initialized to 1. INDEX is used as the subscript for GRADE, the multiply-occurring input field.
- LDATE and SDATE define 6-byte alphanumeric fields that are initialized to blanks. In type 7 procedure logic, CA Culprit moves either the employee's termination date or a blank character string to LDATE. CA Culprit also moves either the employee's start date or a blank character string to SDATE.
- MESSAGE defines a 20-byte alphanumeric field; the field is initialized to blanks. CA Culprit moves literal values to the field in type 7 procedure logic.
- SALGRADE defines a numeric work field initialized to 0. CA Culprit moves values of the multiply-occurring field to SALGRADE in type 7 procedure logic.

Process Parameters

The type 7 procedure logic for Report 01 consists of three chapters. The first chapter of logic determines the employee's employment status; CA Culprit compares FINISH-DATE to the literal '000000', which is the value assigned to active employees. If the employee is no longer active, CA Culprit moves 'PAST EMPLOYEE' to MESSAGE, FINISH-DATE to LDATE, and 0 to SALGRADE. CA Culprit then executes a TAKE, which extracts detail lines 1 and 2 for output. Otherwise, the procedure logic branches to the type 7 process statement that has sequence number 100.

In the second chapter of type 7 procedure logic, CA Culprit processes input records that contain data on active employees. CA Culprit reinitializes LDATE to blanks and compares the employee's salary level to one of four possible levels assigned to the employee's job title.

If the employee's salary is not equal to GRADE.INDEX, CA Culprit increments the value of INDEX by 1. If the value of INDEX is less than or equal to 4, CA Culprit branches to the process parameter that has sequence number 110. This loop continues until either SALARY equals GRADE.INDEX or the value of INDEX exceeds 4.

If the value of SALARY is equal to an occurrence of GRADE, the procedure logic branches to sequence number 400. Otherwise, the procedure logic branches to sequence number 300 once the value of INDEX exceeds 4. Starting with sequence number 300, CA Culprit moves an error message to MESSAGE, reinitializes INDEX and SALGRADE, and extracts detail lines 1 and 2 for output.

The third portion of type 7 procedure logic begins at sequence number 400. This portion of procedure logic processes input buffers of employees with a salary level equal to one of four possible levels for the employee's job title.

If the employee earns the highest amount possible for the job (that is, INDEX equals 4), CA Culprit moves 'AT HIGHEST LEVEL' to MESSAGE, reinitializes SALGRADE and INDEX, and executes a TAKE; otherwise, the procedure logic branches to sequence number 500. Beginning with sequence number 500, CA Culprit moves 'EARNING POTENTIAL' to MESSAGE, increments INDEX by 1, moves the value of GRADE.INDEX to SALGRADE, and releases detail lines 1 and 2 for extracting.

Following the RELS statement, CA Culprit deselects detail line 1 for extracting. The procedure logic moves blanks to SDATE and MESSAGE and branches to the type 7 parameter that has sequence number 510. Within this loop, CA Culprit increments the value of INDEX, compares its value to 4 (the number of occurrences of GRADE), and releases detail line 2 for extracting. Once the value of INDEX exceeds 4, the procedure logic branches to sequence number 600, where INDEX is reinitialized and the input buffer is dropped.

CA Culprit Parameters for Employee Earning Potential Report:

```
mm/dd/yy                  SEQUENTIAL PARAMETER LISTING              Vnn.n  PAGE 1
00 ** SYSIN **                           USE JOBRECS
01 JOBRECS                                IN  80
   INSTALLATION SECURITY OPTION IS NO
                                          REC EMP-ID          1  4  2    'EMPLOYEE ID'
                                          REC START-DATE      6  6       'START DATE'
                                          REC FINISH-DATE    12  6       'TERMINATION' 'DATE'
                                          REC SALARY         18  2  2
                                          REC TITLE          20 20
                                          REC NO-POSITIONS   40  3  2    'NUMBER' 'OF POSTIONS'
                                          REC NO-OPEN        43  3  2    'NUMBER' 'POSITIONS OPEN'
                                          REC SALARY-GRADE   46         GROUP AA 2.4
                                          REC GRADE           1  2  2    ELMNT AA
00 ** SYSIN **                           01OUT   D
                                          013EMPLOYEE EARNING POTENTIAL
                                          01SORT TITLE
                                          01410010 'EMPLOYEE ID'
                                          01410035 'JOB TITLE'
                                          01410057 'SALARY LEVEL'
                                          01420015 'EMPLOYMENT DATES'
                                          01420035 'JOB STATUS'
                                          01430001 ' '
                                          015100100EMP-ID      FN
                                          01510035 TITLE
                                          01510057 SALARY
                                          01520015 SDATE       SZ=6
                                          01520022 LDATE       SZ=6
                                          01520035 MESSAGE     SZ=20
                                          01520057 SALGRADE    SZ=2
                                          010 INDEX  1 LDATE '      ' MESSAGE   '                    '
                                         *    SALGRADE 0   SDATE '      '
                                          017    MOVE START-DATE TO SDATE
                                          017    FINISH-DATE EQ '000000'   100
                                          017    MOVE FINISH-DATE TO LDATE
                                          017    MOVE 'PAST EMPLOYEE ' TO MESSAGE
                                          017    MOVE 0 TO SALGRADE
                                          017    TAKE
                                          017100 MOVE ' ' TO LDATE
                                          017110 SALARY EQ GRADE.INDEX      400
                                          017    INDEX ADD 1 INDEX
                                          017    INDEX GT 4                 300
                                          017    B 110
                                          017300 MOVE 'CHECK SALARY LEVEL' TO MESSAGE
                                          017    MOVE 0 TO SALGRADE
                                          017    MOVE 1 TO INDEX
                                          017    TAKE
                                          017400 INDEX NE  4                 500
                                          017    MOVE 'AT HIGHEST LEVEL' TO MESSAGE
                                          017    MOVE 0 TO SALGRADE
                                          017    MOVE 1 TO INDEX
                                          017    TAKE
                                          017500 MOVE 'EARNING POTENTIAL' TO MESSAGE
                                          017510 INDEX + 1 INDEX
                                          017    INDEX GT 4                  600
                                          017    MOVE GRADE.INDEX TO SALGRADE
                                          017    RELS
                                          017    UNPICK 1
                                          017    MOVE ' ' TO SDATE
                                          017    MOVE ' ' TO MESSAGE
                                          017    B 510
                                          017600 MOVE 1 TO INDEX
                                          017    DROP
```

Employee Earning Potential Output:

```
REPORT NO. 01              EMPLOYEE EARNING POTENTIAL              mm/dd/yy PAGE   1
          EMPLOYEE ID             JOB TITLE             SALARY LEVEL
               EMPLOYMENT DATES   JOB STATUS

          0106                    AR CLERK              12
               800816             EARNING POTENTIAL     14
                                                        15

          0049                    COMPUTER OPERATOR     21
               790929 810401      PAST EMPLOYEE

          0371                    CUMULUS CARETAKER     53
               770304             EARNING POTENTIAL     55

          0048                    DATA ENTRY CLERK      11
               820526 821103      PAST EMPLOYEE

          0471                    DIR WEATHER           82
               820101             EARNING POTENTIAL     84

          0035                    DOCUMENTATION SPEC    43
               800909             AT HIGHEST LEVEL

          0015                    MGR BRAINSTORMING     72
               800102             EARNING POTENTIAL     75

          0015                    MGR BRAINSTORMING     71
               780102 800101      PAST EMPLOYEE

          0471                    MGR BRAINSTORMING     72
               780907 811231      PAST EMPLOYEE

          0013                    MGR PERSONNEL         72
               810102             EARNING POTENTIAL     73
                                                        75

          0349                    MGR THERMOREGULATION  72
               791111             AT HIGHEST LEVEL

          0119                    PHOTOGRAPHER          33
               771214             EARNING POTENTIAL     38
                                                        39

          0149                    PR WRITER             33
               770908             EARNING POTENTIAL     34

          0030                    PRESIDENT             93
               731121             EARNING POTENTIAL     95

          0023                    PROGRAMMER TRAINEE    43
               780504 790504      PAST EMPLOYEE

          0021                    PROGRAMMER TRAINEE    21
               801221             EARNING POTENTIAL     41
                                                        42
                                                        43

          0023                    PROGRAMMER/ANALYST    44
```

Appendix I: Sample Programs—Database Access

This section contains the following topics:

About This Appendix

This appendix contains three sample CA Culprit programs that access data in a non-SQL defined database using either database records or a logical record:

- The first program navigates the DEPARTMENT, EMPLOYEE, and OFFICE records of the database and processes the data depending upon the value of PATH-ID in the input buffer.
- The second program navigates a bill-of-materials set relationship within the database and executes DB-EXIT calls within the procedure logic of the report to retrieve database records.
- The third program specifies a logical record on the PATH parameter. CA Culprit processes logical records according to the selection criteria specified on the WHERE clause in the PATH parameter.

Appendix K: Employee Database Subschema contains the structure of the Employee database and the definitions of the fields that are used as input to these programs. Each program is discussed separately below.

Example 1—Employee location and status by department

This program reads the DEPARTMENT, EMPLOYEE, and OFFICE database records. For each department that contains employees, the output lists the ID and name of each employee, the employee's office location, and the status of the employee in terms of disability, leave of absence, or termination.

The program performs the following functions:

- Sorts the information within each department by location and within each location by employee last name
- Begins printing a new page each time the name of the department changes
- Tests the value of PATH-ID for alternate path ID values specified on the PATH parameter
- Prints messages depending upon the value of PATH-ID and the employee status field

The first figure shows the CA Culprit code that created the report shown in the second figure.

Parameters for Employee Location and Status by Department:

```
DATABASE DICTNAME=DOCANWK
IN DB SS=EMPSS01,EMPSCHM,100
PATHAA DEPARTMENT EMPLOYEE(BB) OFFICE(CC)
010 MESSAGE '                    '
010 TERM-DATE '      '
010 EMP-ID 0000
01OUT D
01SORT DEPT-NAME-0410,1 OFFICE-CODE-0450 EMP-LAST-NAME-0415
013EMPLOYEE LOCATION AND STATUS BY DEPARTMENT
0141*010 DEPT-NAME-0410
0151*010 EMP-ID      SZ=4     FN   HH 'EMPLOYEE ID'
0151*020 EMP-NAME-0415             HH 'EMPLOYEE NAME'
0151*030 OFFICE-CITY-0450          HH 'LOCATION'
0151*040 MESSAGE                   HH 'STATUS'
0151*050 TERM-DATE
0152*010 EMP-ID      SZ=4     FN
0152*020 EMP-NAME-0415
0152*040 MESSAGE
0153*010 MESSAGE
017      PATH-ID NE 'BB'           050
017      MOVE 'NO EMPLOYEES' TO MESSAGE
017      TAKE 3
017050   EMP-ID-0415 NE EMP-ID     100
017      DROP
017100   MOVE EMP-ID-0415 TO EMP-ID
017      PATH-ID NE 'CC'           200
017      MOVE 'NO OFFICE' TO MESSAGE
017      TAKE 2
017200   IF STATUS-0415 NE '05'    300
017      MOVE 'TERMINATED' TO MESSAGE
017      MOVE TERMINATION-DATE-0415 TO TERM-DATE
017      B   700
017300   IF STATUS-0415 NE '04'    400
017      MOVE 'LEAVE-OF-ABSENCE' TO MESSAGE
017      B   625
017400   IF STATUS-0415 NE '03'    500
017      MOVE 'LONG-TERM-DISABILITY' TO MESSAGE
017      B   625
017500   IF STATUS-0415 NE '02'    600
017      MOVE 'SHORT-TERM-DISABILITY' TO MESSAGE
017      B   625
017600   MOVE ' ' TO MESSAGE
017625   MOVE ' ' TO TERM-DATE
017700   TAKE 1
```

Report for Employee Location and Status by Department:

```
REPORT NO. 01      EMPLOYEE LOCATION AND STATUS BY DEPARTMENT                mm/dd/yy PAGE   1
   ACCOUNTING AND PAYROLL
                   EMPLOYEE ID                          EMPLOYEE NAME            LOCATION            STATUS

                       2002                       ROCCO     COLOMBO          SPRINGFIELD
                       0069                       JUNE      BLOOMER          BOSTON
                       0100                       EDWARD    HUTTON           BOSTON
                       0011                       RUPERT    JENSON           BOSTON
                       0067                       MARIANNE  KIMBALL          BOSTON
REPORT NO. 01         EMPLOYEE LOCATION AND STATUS BY DEPARTMENT                  mm/dd/yy PAGE   1
   BRAINSTORMING
                   EMPLOYEE ID                          EMPLOYEE NAME            LOCATION            STATUS

                       0334                       CAROLYN   CROW             GLASSTER
                       0301                       BURT      LANCHESTER       GLASSTER
                       0015                       RENE      MAKER            GLASSTER
                       0466                       BAY       ANDOVER          WESTON            LONG-TERM-DISABILITY
                       0467                       S.E.A.    BREEZE           WESTON
                       0457                       HARRY     HARM             WESTON            TERMINATED            091883
                       0341                       RICHARD   MUNYON           WESTON
                       0458                       RICHARD   WAGNER           WESTON
REPORT NO. 01        EMPLOYEE LOCATION AND STATUS BY DEPARTMENT                 mm/dd/yy PAGE   6
   CLIMBING
                   EMPLOYEE ID                          EMPLOYEE NAME            LOCATION            STATUS

               NO EMPLOYEES
REPORT NO. 01       EMPLOYEE LOCATION AND STATUS BY DEPARTMENT         mm/dd/yy PAGE   19
   PUBLIC RELATIONS
                   EMPLOYEE ID                          EMPLOYEE NAME            LOCATION            STATUS

                       0476                       BETSY     ZEDI             SPRINGFIELD       TERMINATED            072284
                       0120                       MICHAEL   ANGELO           BOSTON
                       0007                       DONTE     BANK             BOSTON
                       0158                       JOCK      JACKSON          BOSTON
                       0127                       CAROL     MCDOUGALL        BOSTON
                       0149                       LAURA     PENMAN           BOSTON
```

Note: Information for each department prints on a separate page.

Parameters coded for this report are described below:

DATABASE and INPUT Parameters

The DATABASE and INPUT parameters define the input for a database access run. The INPUT parameter specifies DB, which indicates an IDMS/R database by default. The SS= keyword expression identifies the subschema, or view, of the database to be accessed. The dictionary named on the DATABASE parameter, DOCANWK, contains the schema and subschema definition. In a CV environment, DOCANWK must also contain the EMPSS01 load module.

PATH Parameter

The PATH parameter specifies three records:

- The DEPARTMENT record is the entry record for the run. CA Culprit retrieves each DEPARTMENT record occurrence as it is encountered in the database.
- The EMPLOYEE record is the member record of the DEPT-EMPLOYEE set relationship. Path ID BB indicates that no EMPLOYEE record occurrences participate in the set relationship with a particular DEPARTMENT record occurrence.
- The OFFICE record is the owner record of the OFFICE-EMPLOYEE set relationship. Path ID CC indicates that no OFFICE record occurrences participate in the set relationship with a particular EMPLOYEE record occurrence.

Work Field Parameters

Report 01 specifies two alphanumeric work fields and one numeric work field:

- MESSAGE is an alphanumeric work field initialized as a 20-blank literal. MESSAGE serves as the result field for MOVE operations in the procedure logic for Report 01 and as an output field on a type 5 edit parameter.
- TERM-DATE is an alphanumeric work field that has an initial value of six blank spaces. TERM-DATE receives the value of an employee's termination date if the status code for the employee is 05; otherwise, TERM-DATE is reinitialized to blanks.
- EMP-ID is a numeric work field initialized to zero. CA Culprit compares the value in this field to the value in the employee ID field in the EMPLOYEE record for the purpose of removing duplicate employee entries.

Output Definition Parameters

The OUTPUT parameter for Report 01 defines a printed report that contains details-only information and 132 characters per line. The SORT parameter indicates that the information for the report is sorted by department name. The sort break associated with the department name instructs CA Culprit to begin printing a new page each time the name changes. Within each department, the information is sorted by the last name of each employee within the same office.

Since DEPT-NAME-0410 appears on the SORT parameter, it can be specified on a type 4 edit parameter. CA Culprit prints the department name below the report title at the top of each page. CA Culprit then prints the automatic headers defined on the type 5 edit parameters for the report.

The type 5 edit parameters define three detail lines:

- Detail line 1 contains the employee id, employee name, office location, and a message, depending on whether the employee is actively employed. If the employee has left the company, the detail line also contains the date the employee stopped working.
- Detail line 2 contains the employee id, employee name, and a message; the line is printed if an employee does not have an office.
- Detail line 3 specifies only the MESSAGE field; the line is printed only if a department has no employees.

Process Parameters

The procedure logic for Report 01 contains a number of comparisons to determine the actual contents of the input buffer:

- The first comparison determines whether the value of PATH-ID is BB. If the path ID is BB, CA Culprit was not able to retrieve any EMPLOYEE record occurrences for the current DEPARTMENT record. The procedure logic instructs CA Culprit to move 'NO EMPLOYEES' to MESSAGE and print detail line 3.
- The second comparison beginning at sequence number 050 determines whether the employee ID field contains a duplicate value; if it does, CA Culprit stops processing the input buffer.
- The third comparison determines whether the value of PATH-ID is CC. If the path ID is CC, CA Culprit moves a literal to MESSAGE and prints detail line 2. The Run Time Messages listing for this report indicates that CA Culprit did not return any strings for path CC, as shown below:

```
STRINGS RETURNED FOR PATH AA-     81
STRINGS RETURNED FOR PATH BB-     11
STRINGS RETURNED FOR PATH CC-      0
```

- The procedure logic that begins with sequence number 200 pertains to a complete input buffer; that is, when the value of PATH-ID is AA. Depending upon the value of the STATUS-0415 field, CA Culprit moves various literals to MESSAGE and prints detail line 1.

Example 2—Titles and Skills By Department

CA Culprit enters the database by using the CALC-key value for the DEPARTMENT record specified on a KEY parameter. CA Culprit then navigates a bill-of-materials structure that returns the manager and staff members within the department.

The first figure shows the CA Culprit code that created the report shown in the scond figure.

Parameters for Department Brainstorming:

```
 DATABASE DICTNAME=DOCANWK
 IN DB SS=EMPSS01,EMPSCHM,100
 KEY DEPT-ID-0410 5100
 PATHAA DEPARTMENT EMPLOYEE STRUCTURE (MANAGES)
*       EMPLOYEE (REPORTS-TO)
 PATH-- EXPERTISE SKILL EMPOSITION JOB
 SELECT EMPLOYEE 2 IN PATH AA WHEN STATUS-0415 EQ '01'
 010UT 100 D
 01SORT EMP-NAME-0415 (1)
 013MANAGER AND STAFF OF DEPARTMENT BRAINSTORMING
 0141*0010'MANAGER'
 0141*010 EMP-NAME-0415 (1)
 0142*020 'TITLE AND'
 0143*010 'STAFF'
 0143*020 'SKILL LEVEL'
 0143*030 'SKILL'
 0151*0100EMP-NAME-0415 (2)
 0151*020 TITLE-0440
 0152*020 SKILL-LEVEL-0425
 0152*030 SKILL-NAME-0455
 017      CALL DB-EXIT ('NEXT' 'EMPOSITION ' 'EMP-EMPOSITION ')
 017      IF IDMS-STATUS NE '0000' DROP
 017      CALL DB-EXIT ('OWNER' 'JOB ' 'JOB-EMPOSITION ')
 017      IF IDMS-STATUS NE '0000' DROP
 017      RELS 1
 017010   CALL DB-EXIT ('NEXT' 'EXPERTISE ' 'EMP-EXPERTISE ')
 017      IF IDMS-STATUS EQ '0307' TAKE
 017      IF IDMS-STATUS NE '0000' DROP
 017      CALL DB-EXIT ('OWNER' 'SKILL ' 'SKILL-EXPERTISE ')
 017      IF IDMS-STATUS NE '0000' DROP
 017      RELS 2
 017      B  010
```

Report for Department Brainstorming:

```
REPORT NO. 01            MANAGER AND STAFF OF DEPARTMENT BRAINSTORMING       mm/dd/yy PAGE     1

      MANAGER       RENE      MAKER
                                                    TITLE AND
                              STAFF                 SKILL LEVEL             SKILL

                    S.E.A.    BREEZE                KEEPER OF THE WINDS
                                                          04                WIND

                    RAY       ANDOVER               SNOWBLOWER
                                                          04                LIGHTNING
                                                          02                THUNDER
                                                          02                HURRICANE
                                                          02                TORNADO
                                                          02                TIDAL WAVE

                    RICHARD   WAGNER                STURM/DRANG ADMIN
                                                          04                HURRICANE
                                                          04                OPERA
                                                          04                THUNDER

                    HARRY     HARM                  STURM/DRANG ADMIN
                                                          04                THUNDER

                    RICHARD   MUNYON                SNOWBLOWER
                                                          04                WIND
                                                          04                SNOW
                                                          03                COLD

                    CAROLYN   CROW                  RAINDANCE CONSULTANT
                                                          04                RAIN
                                                          04                CHOREOGRAPHY

                    BURT      LANCHESTER            RAINMAKER
                                                          04                DRIZZLE
                                                          04                RAIN
                                                          03                FOG

  RECORDS WRITTEN FOR REPORT 01—       29
```

Parameters coded for the report are described below:

DATABASE and INPUT Parameters

The DATABASE and INPUT parameters for this CA Culprit run are the same as defined in Example 1 earlier in this chapter.

KEY Parameter

The KEY parameter limits the number of DEPARTMENT records to be processed in the CA Culprit run to 1. The parameter identifies the CALC-key field of the DEPARTMENT record, DEPT-ID-0410, and a numeric value, 5100. CA Culprit directly accesses the DEPARTMENT record occurrence that has 5100 for the DEPT-ID-0410 field and continues to navigate the database according to the route defined on the PATH parameter.

PATH Parameters

The CA Culprit code contains an automatic retrieval and a dummy PATH parameter. Each input buffer that is processed in the CA Culprit run contains an occurrence of the records that define path AA. The EMPLOYEE STRUCTURE (MANAGES) EMPLOYEE (REPORTS-TO) portion of path AA defines a bill-of-materials structure in which the first EMPLOYEE record occurrence contains the manager of the employee named in the second EMPLOYEE record occurrence.

Each input buffer also reserves enough space to accommodate the records named on the dummy PATH parameter, identified by two hyphens (--). CA Culprit retrieves occurrences of these records when it encounters a call to DB-EXIT.

SELECT Parameter

The SELECT parameter applies selection criteria to the second EMPLOYEE record in path AA. If the EMPLOYEE record fails the selection criteria, the input buffer is not processed. In this example, CA Culprit selects all occurrences of the second EMPLOYEE record that specify 01 for the employee status; this value indicates that the employee is actively employed.

Output Definition Parameters

The output for Report 01 is a printed report that contains details-only information. The line size for the report is 100 characters.

The SORT parameter specifies a field associated with the first EMPLOYEE record in the input buffer since the field name is qualified by level number 1. In this example, the SORT parameter is required because EMP-NAME-0415 is specified on a header line.

Report 01 contains a title and three header lines. A spacing code associated with header line 1 generates a blank line between the title and the headings. Report 01 also specifies two types of detail lines. The first contains the name and title of an employee that reports to the manager named in the heading; the second identifies the skills and the skill levels that the staff member has mastered.

Process Parameters

The procedure logic issues four calls to DB-EXIT in order to retrieve the job title, skill level, and skills of each staff member. After each call, the procedure logic tests the value of the IDMS-STATUS field to determine whether the database record was retrieved successfully, as follows:

- The first DB-EXIT call retrieves an occurrence of the EMPOSITION record that is associated with the second EMPLOYEE record occurrence currently in the input buffer. CA Culprit stops processing the input buffer if the IDMS-STATUS code for the call is anything other than 0000.
- The second DB-EXIT call retrieves an occurrence of the JOB record that is the owner of the EMPOSITION record occurrence. The JOB record contains the field TITLE-0440, which contains job title values. Once CA Culprit retrieves the JOB record, it prints detail line 1.
- The third DB-EXIT call is similar to the first, except in this case, CA Culprit retrieves an occurrence of the EXPERTISE record that relates to the second EMPLOYEE record occurrence in the input buffer. Since an employee may have more than one skill level, identified as the field SKILL-LEVEL-0425, CA Culprit iterates through the procedure logic until it encounters an end-of-set condition, identified as status code 0307.
- The fourth DB-EXIT call is similar to the second. CA Culprit retrieves an occurrence of the SKILL record that is the owner of the EXPERTISE record. The SKILL record contains the field SKILL-NAME-0455, which contains the names of various skills. Provided the IDMS-STATUS code is 0000, CA Culprit prints detail line 2 and branches to the process statement that issues a DB-EXIT call for the next EXPERTISE record occurrence.

Example 3—Job Titles of Company Employees

The input for this example is a logical record that contains the OFFICE, DEPARTMENT, EMPLOYEE, and JOB records shown in Employee Database Subschema. CA Culprit accesses occurrences of the logical record by using key values that correspond to the CALC-key field of the JOB record. The output for the report lists all employees that share the same job id.

CA Culprit performs the following functions in this program:

- Accesses a key file that contains values that correspond to the JOB-ID-0440 field in the logical record
- Selectively retrieves logical record occurrences that meet the criteria specified in the WHERE clause on the PATH parameter
- Sorts the data by department name
- Prints the name and location of all employees that share the same job below a detail line that contains the department name associated with the job, the job id, and the job title

The first figure shows the CA Culprit parameters that produced the report shown in the second figure.

Parameters for Employee Job Titles:

```
 DATABASE DICTNAME=DOCANWK
 IN DB SS=EMPSS09,EMPSCHM,100
 KEYFILE 80 KF=1 4 LRFNAME=JOB-ID-0440 DD=SYS002
 PATHAA EMP-JOB-LR WHERE JOB-ID-0440 EQ KEY-VALUE AND
*         (DEPT-NAME-0410 EQ 'COMPUTER OPERATIONS'
*       OR DEPT-NAME-0410 EQ 'INTERNAL SOFTWARE'
*       OR DEPT-NAME-0410 EQ 'BRAINSTORMING'
*       OR DEPT-NAME-0410 EQ 'BLUE SKIES')
 01OUT  D
 01SORT DEPT-NAME-0410
 013JOB TITLES OF COMPANY EMPLOYEES
 0151*0100DEPT-NAME-0410            HH  ' ' 'DEPARTMENT' 'AND'
*                                       'EMPLOYEE NAME'
 0151*020 JOB-ID-0440    SZ=4 FN    HH 'JOB ID' 'AND' 'OFFICE'
 0151*030 TITLE-0440                HH 'TITLE'
 0152*010 EMP-NAME-0415
 0152*020 OFFICE-CITY-0450
 010     JOB-ID 0000
 017100  IF JOB-ID-0440 EQ JOB-ID  200
 017     MOVE JOB-ID-0440 TO JOB-ID
 017     RELS 1
 017200  RELS 2
```

Report for Employee Job Titles:

```
REPORT NO. 01        JOB TITLES OF COMPANY EMPLOYEES       mm/dd/yy PAGE     1

                     DEPARTMENT                          JOB ID
                        AND                               AND
                    EMPLOYEE NAME                        OFFICE              TITLE


    BLUE SKIES                                            5037          SUNSHINE SUPERVISOR
              ALAN      DONOVAN                       GLASSTER

    BLUE SKIES                                            5039          CUMULUS CARETAKER
              BETH  M.  CLOUD                         GLASSTER

    BLUE SKIES                                            5005          MGR BLUE SKIES
              DANIEL    MOON                          GLASSTER

    BRAINSTORMING                                         5025          SNOWBLOWER
              RICHARD   MUNYON                        WESTON

    BRAINSTORMING                                         5001          MGR BRAINSTORMING
              RENE      MAKER                         GLASSTER

    BRAINSTORMING                                         5027          KEEPER OF THE WINDS
              S.E.A.    BREEZE                        WESTON

    BRAINSTORMING                                         5029          STURM/DRANG ADMIN
              RICHARD   WAGNER                        WESTON

    BRAINSTORMING                                         5021          RAINMAKER
              BURT      LANCHESTER                    GLASSTER

    COMPUTER OPERATIONS                                   3051          DATA ENTRY CLERK
              SANDY     KRAAMER                       BOSTON
              ROBIN     GARDNER                       BOSTON
              GEORGE    FONRAD                        BOSTON

    COMPUTER OPERATIONS                                   3003          MGR COMPUTER OPS
              HERBIE    BABIE                         SPRINGFIELD

    COMPUTER OPERATIONS                                   3029          COMPUTER OPERATOR
              JANE      FERNDALE                      SPRINGFIELD
              HERBERT   LIPSICH                       SPRINGFIELD

    COMPUTER OPERATIONS                                   3023          SYSTEMS PROGRAMMER
              TERENCE   KLWELLEN                      SPRINGFIELD

    INTERNAL SOFTWARE                                     3027          PROGRAMMER TRAINEE
              NGUSA     KARL-I-BOND                   SPRINGFIELD

    INTERNAL SOFTWARE                                     3011          DATABASE ADMIN.
              JAMES     JACOBI                        SPRINGFIELD

    INTERNAL SOFTWARE                                     3001          MGR INTERNL SOFTWARE
              JENNIFER  GARFIELD                      SPRINGFIELD

    INTERNAL SOFTWARE                                     3025          PROGRAMMER/ANALYST
              JAMES     PAKKER                        SPRINGFIELD
              PERCY     EINSTEIN                      SPRINGFIELD
              VLADIMIR  HEAROWITZ                     SPRINGFIELD
              JANE      SOUGH                         SPRINGFIELD
              KATHERINE O'HEARN                       SPRINGFIELD
              JULIE     JANSSEN                       SPRINGFIELD

  RECORDS WRITTEN FOR REPORT 01—       52
```

The parameters coded for this report are described below:

DATABASE and INPUT Parameters

The DATABASE parameter for this run is the same as described for the other examples in this chapter. The INPUT parameter is also the same, except that in this case, the subschema to be accessed is EMPSS09.

KEYFILE Parameter

The code also specifies a KEYFILE parameter, which identifies a sequential file that contains fixed-length 80-byte records. The key-file field begins in column 1 for a length a four bytes; the values in this field are to be compared to the logical record field identified in the LRFNAME= keyword expression on the KEYFILE parameter.

PATH Parameter

The PATH parameter specifies logical record EMP-JOB-LR, followed by a WHERE clause that limits the number of logical record occurrences CA Culprit retrieves, as follows:

- CA Culprit retrieves logical record occurrences that specify a value for JOB-ID-0440 that equals the current value in the key-file field.
- Of the records that pass the first selection criteria, CA Culprit selects all record occurrences that specify either COMPUTER OPERATIONS, INTERNAL SOFTWARE, BRAINSTORMING, or BLUE SKIES for a department name.

Work Field Definition Parameter

Report 01 specifies a numeric work field, JOB-ID, which is initialized to zero. CA Culprit compares the value of JOB-ID-0440 in the logical record to the value of the work field in order to eliminate duplicate occurrences.

Output Definition Parameters

Report 01 is a printed report that contains details-only information. Following the title, CA Culprit prints headings at the top of each report page from the auto-headers specified on the type 5 edit parameters. To generate a blank line between the title and the headings, the first auto-header literal associated with field DEPT-NAME-0410 contains a single blank.

The detail lines of the report are sorted in alphabetical order by the name of the department associated with each job. The first detail line outputs the name of the department, the job id, and the job title in relative columns *010, *020, and *030, respectively. The spacing code associated with detail line 1 instructs CA Culprit to skip a line before it outputs the data.

The second detail line outputs the name of the employee and the employee's job location in relative columns *010 and *020. CA Culprit prints detail line 2 as long as the value of the JOB-ID-0440 field remains the same.

Process Parameters

The procedure logic for Report 01 compares the value of the JOB-ID-0440 field in the input buffer to the value in the JOB-ID work field. When the value in the input buffer changes, CA Culprit moves the new value to the work field and prints detail lines 1 and 2. If the value remains the same, CA Culprit branches to process statement 200 and prints detail line 2.

Appendix J: Sample Programs—Table Access and Creation

This section contains the following topics:

Overview

This appendix contains four sample CA Culprit programs that access or create non-SQL defined data tables:

- The first program obtains input from a sequential file that contains information about company employees. The output creates two tables and replaces the information contained in a third.
- The second program navigates the database structure shown in Employee Database Subschema. The output from the CA Culprit run produces two reports; one report adds new information to a previously defined table and the second report creates a new table.
- The third program uses a table as input to the CA Culprit run. One report prints the contents of the table and a second report creates another table.
- The fourth program consolidates three tables and prints the contents of the tables.

Each CA Culprit program is discussed separately below.

Example 1—Creating Tables with Sequential Files

This program reads a sequential file that contains employee-related information. As shown in the first figure, the INPUT parameter characterizes a sequential file that contains fixed-length 200-byte records. The input fields are defined in library member RRECS, which CA Culprit copies at run time; the second figure illustrates the contents of RRECS.

Parameters Coded for Example 1:

```
 DATABASE DICTNAME=ASFDICT
 PROFILE USER=JFD PW=ROSEBUD
 IN 200
USE RRECS
 01$
 01$           CREATE TABLE EMPLOYEE TABLE
 01$
 01OUT TABLE='EMPLOYEE TABLE' TYPE=CREATE ONLINE=Y
 0151*010 EMP-NAME    SZ=25  IX=A
 0151*020 STREET      SZ=20
 0151*030 CITY        SZ=15
 0151*040 STATE       SZ=2
 0151*050 ZIP-CODE    SZ=5
 0151*060 PHONE            FM '999-999-9999'
 0151*070 START-DATE       FD
 02$
 02$           REPLACE TABLE STATUS
 02$
 02OUT TABLE=STATUS TYPE=REPLACE
 020 DISABILITY '                          '
 0251*010 EMP-NAME     SZ=25
 0251*020 TITLE        SZ=20
 0251*030 DISABILITY   SZ=25
 027    IF STATUS EQ '02' 300
 027    IF STATUS EQ '03' 400
 027    DROP
 027300 MOVE 'SHORT-TERM DISABILITY' TO DISABILITY
 027    TAKE
 027400 MOVE 'LONG-TERM DISABILITY' TO DISABILITY
 027    TAKE
 03$
 03$           CREATE TABLE 'DEPARTMENT SALARIES'
 03$
 03OUTPUT T TABLE='DEPARTMENT SALARIES' TYPE=CREATE CATALOG=TSTDICT
*           ONLINE=YES
 03SORT DEPT-NAME,+
 0351*020 SALARY       SZ=10  FP
 0351*000 COUNT        SZ=4   FP
 0361*010 DEPT-NAME    SZ=30
 0361*020 SALARY       SZ=10  FP
 0361*030 COUNT        SZ=4   FP
 0361*040 AVERAGE      SZ=10  FP
 030 COUNT 1
 030 AVERAGE DP=2  0
 038010   LEVL EQ 2  DROP
 038      SALARY DIVIDE COUNT  AVERAGE
```

Input Field Definitions for Example 1:

```
REC EMP-ID          1  4  2
REC FIRST-NAME      5  10
REC LAST-NAME      15  15
REC EMP-NAME        5  25      'EMPLOYEE' 'NAME'
REC STREET         30  20      'STREET'
REC CITY           50  15      'CITY'
REC STATE          65   2      'STATE'
REC ZIP-CODE       67   5      'ZIP'
REC PHONE          72  10 2    'PHONE'
REC STATUS         82   2      'STATUS'
REC SOC-SEC-NUM    84   9 2    'SOCIAL' 'SECURITY' 'NUMBER'
REC START-DATE     93   6 2    'STARTING DATE'
REC LEAVE-DATE     99   6 2    'LEAVING DATE'
REC BIRTH-DATE    105   6 2    'BIRTHDAY'
REC DEPT-ID       111   4 2
REC DEPT-NAME     115  45
REC SALARY        160   7 3 DP=2
REC JOB-ID        167   4 2
REC TITLE         171  20      'JOB TITLE'
```

The code that appears in the first figure generates three reports that create and populate two tables and replace information in a third. The code for each report is discussed separately below.

Creating EMPLOYEE TABLE

Report 01 specifies an OUTPUT parameter that instructs CA Culprit to create EMPLOYEE TABLE and define it to ASFDICT. Because the OUTPUT parameter includes ONLINE=Y, JFD and other authorized users can access and modify the information in EMPLOYEE TABLE online by using ASF.

As shown in the following figure, each row in EMPLOYEE TABLE contains the name, address, phone number, and hire date of a company employee. Column EMP-NAME specifies IX=A, which establishes an index for the table. The information stored in columns PHONE and START-DATE is formatted according to the options specified on the type 5 edit parameters. CA Culprit stores the formatted numeric values as text fields.

A Portion of Table EMPLOYEE TABLE:

EMP-NAME		STREET	CITY	STATE	ZIP-CODE	PHONE	START-DATE
ELEANOR	PEOPLES	756 YELLOWSTONE DR	BOSTON	MA	02834	617-329-1212	01/02/81
ALAN	DONOVAN	6781 CORNWALL AVE	MELROSE	MA	02176	617-665-5412	10/10/81
ALBERT	BREEZE	100 BOARDWALK	OCEAN CITY	NJ	03461	617-554-2387	06/02/79
BETH M.	CLOUD	3456 PINKY LN	NATICK	MA	02178	617-432-1212	03/04/77
BETSY	ZEDI	34 VALE AVE	SOUTHBORO	MA	03145	617-431-9909	02/23/76
BRIAN	NICEMAN	60 FLORENCE AVE	MELROSE	MA	02176	617-665-4315	05/06/80
BURT	LANCHESTER	45 PINKERTON AVE	WALTHAM	MA	01476	617-534-1109	02/03/75
CAROL	MCDOUGALL	19 URITOP DR	WELLESLEY	MA	01568	617-887-1324	06/07/80
CAROLYN	CROW	891 SUMMER ST	WESTWOOD	MA	02090	617-329-1776	06/17/79

Note: The column headings are the names of the input fields.

Replacing STATUS

Table STATUS contains the names and titles of employees who are on short- or long-term disability. The OUTPUT parameter instructs CA Culprit to replace the information in the existing table with current data.

Because the OUTPUT parameter specifies REPLACE, the order, size, and format of the output fields must be the same as those specified when STATUS was created. The procedure logic for Report 02 determines the employee's status code; if the code is either 02 or 03, CA Culprit moves a literal to the DISABILITY field. The contents of the table appear in the following figure. No employees are on long-term disability and only one is on short-term disability.

Contents of Table STATUS:

EMP-NAME	TITLE	DISABILITY
JACK KRAMER	DATA ENTRY CLERK	SHORT-TERM DISABILITY

Creating DEPARTMENT SALARIES

Report 03 creates DEPARTMENT SALARIES, a totals-only table that summarizes the number of employees, total salary, and average salary for each department.

Contents of Table DEPARTMENT SALARIES:

DEPT-NAME	SALARY	COUNT	AVERAGE
ACCOUNTING AND PAYROLL	214,500.00	6	35,750.00
BLUE SKIES	158,250.00	3	52,750.00
BRAINSTORMING	262,000.00	5	52,400.00
COMPUTER OPERATIONS	234,750.00	9	26,083.33
EXECUTIVE ADMINISTRATION	510,000.00	4	127,500.00
INTERNAL SOFTWARE	390,500.00	10	39,050.00
MAIL ROOM	182,000.00	4	45,500.00
PERSONNEL	145,500.00	4	36,375.00
PUBLIC RELATIONS	226,000.00	6	37,666.67

The code that generated DEPARTMENT SALARIES is discussed below:

- The type 5 edit parameters write a record to the extracted items and statistics file, which contains the salary for each employee and a counter. Each time a control break occurs, CA Culprit totals the salaries and number of employees for each department.

- The type 6 edit parameters determine the definition of each column in the table and provide a name for a column heading. For example, SALARY is defined as a 10-byte packed decimal field; the heading for the column is SALARY.
- The type 8 process parameters for Report 03 are processed each time the department name changes (LEVL = 1) and at grand totals time (LEVL = 2). Generally, totals-only tables do not include a grand totals line. Therefore, process statement 010 drops all records when LEVL equals 2. When LEVL equals 1, CA Culprit computes the average salary for the current department and outputs it on a total line.

Example 2—Creating Tables from Database Records

This program reads the data from the Employee database that appears in Figure K-1. As shown in the following figure, EMPSS01 is the view of the database to be accessed, and DOCANWK is the name of the dictionary that stores the database definitions. Since the PATH parameter specifies only a primary path id, CA Culprit will navigate the database and return an occurrence of each record that the PATH parameter specifies.

Parameters Coded for Example 2:

```
 DATABASE DICTNAME=DOCANWK
 PROFILE USER=LHN PW=ROSEBUD
 INPUT DB SS=EMPSS01,EMPSCHM,100
 PATHAA JOB EMPOSITION EMPLOYEE DEPARTMENT
 01$
 01$             ADD TO TABLE STATUS
 01$
 010UTPUT TABLE=STATUS TYPE=ADD CATALOG=ASFDICT
 0151*010 EMP-NAME   SZ=25
 0151*020 TITLE      SZ=20
 0151*030 DISABILITY SZ=20
 017      IF STATUS-0415 NE '04' DROP
 017      MOVE EMP-NAME-0415 TO EMP-NAME
 017      MOVE TITLE-0440 TO TITLE
 017      MOVE 'LEAVE-OF-ABSENCE' TO DISABILITY
 010 DISABILITY '                    '
 010 EMP-NAME   '                        '
 010 TITLE      '                    '
 02$
 02$            CREATE TABLE 'JOB SALARIES'
 02$
 020UTPUT T TABLE='JOB SALARIES' TYPE=REPLACE
*           CATALOG=ASFDICT ONLINE=YES
 02SORT JOB-ID,+,TITLE
 0251*030 SALARY
 0251*000 NUMBER-OF-EMPLOYEES
 0261*010 JOB-ID                 FM '9999'
 0261*020 TITLE                  SZ=20
 0261*030 SALARY                 SZ=6  FP
 0261*040 NUMBER-OF-EMPLOYEES    SZ=4  FZ
 020      JOB-ID 0
 020      NUMBER-OF-EMPLOYEES 1
 020      SALARY DP=2 0
 020      TITLE '                    '
 027      MOVE JOB-ID-0440 TO JOB-ID
 027      MOVE SALARY-AMOUNT-0420 TO SALARY
 027      MOVE TITLE-0440 TO TITLE
 028      IF LEVL EQ 2 DROP
```

This program generates two reports: Report 01 adds rows to an existing table and Report 02 creates and populates a table. Each report is discussed separately below.

Adding Rows to STATUS

Table STATUS itemizes the name and job title of employees that are on short- or long-term disability. In Example 1, the rows in STATUS were replaced with rows of current employee information. In this example, the names and titles of employees who are on a leave of absence will be added to the table. As shown in the following figure, CA Culprit adds one row of information to the ...

New Contents of Table STATUS:

EMP-NAME		TITLE	DISABILITY
JACK	KRAMER	DATA ENTRY CLERK	SHORT-TERM DISABILITY
JOE	NGUYA	WINTERIZER	LEAVE-OF-ABSENCE

The OUTPUT parameter instructs CA Culprit to access STATUS, a table owned by LHN and defined in ASFDICT. Since CA Culprit is adding rows to an existing table, the type 5 edit parameters must specify the same field definitions that exist in ASFDICT.

The fields specified on the type 5 edit parameters are work fields because the names of the database fields would make inappropriate column headings. The procedure logic for Report 01 moves the database field values to the work fields once CA Culprit has determined that the employee is on a leave of absence (status code 04).

Creating JOB SALARIES

Report 02 creates table JOB SALARIES, which is owned by LHN and defined in ASFDICT. Each row of the table contains the ID and title of a job within the company, the number of employees that perform the job, and the total salary of these employees.

Contents of Table JOB SALARIES:

JOB-ID	TITLE	SALARY	NUMBER-OF-EMPLOYEES
1001	MGR PERSONNEL	$80,000.00	1
1023	RECRUITER/INTERVWR	$39,000.00	1
1051	PERSONNEL CLERK	$26,500.00	2
2001	MGR ACCTNG/PAYROLL	$82,000.00	1
2023	ACCOUNTANT	$45,000.00	1
2025	FINANCIAL ANALYST	$44,000.00	1
2051	AP CLERK	$14,000.00	1
2053	AR CLERK	$14,500.00	1
2055	PAYROLL CLERK	$15,000.00	1
3001	MGR INTERNL SOFTWARE	$141,000.00	2
3003	MGR COMPUTER OPS	$205,000.00	3
3011	DATABASE ADMIN.	$110,000.00	2
3023	SYSTEMS PROGRAMMER	$43,000.00	1
3025	PROGRAMMER/ANALYST	$213,000.00	6
3027	PROGRAMMER TRAINEE	$58,000.00	2
3029	COMPUTER OPERATOR	$61,000.00	3

The code that created JOB SALARIES is discussed below:

- The SORT parameter specifies a numeric work field, JOB-ID, a break code, and an alphanumeric work field, TITLE. CA Culprit executes a control break each time the value of JOB-ID changes and outputs the value in column *010 on a type 6 edit parameter. The break code instructs CA Culprit to perform total time processing without generating any blank rows in the table.

  The SORT parameter specifies TITLE in order to output the sort-key value on a type 6 edit parameter. If TITLE were not coded on the SORT parameter, CA Culprit would output the current value of TITLE at total time.

- The type 5 edit parameters write a record to the extracted items and statistics file, which contains the salary for each employee and a counter. Each time a control break occurs, CA Culprit totals the salaries and number of employees for each job.

- The type 6 edit parameters establish the headings and column definitions for the table. Since the field names in the database are not as meaningful as column headings, the type 6 edit parameters specify work fields into which CA Culprit moves the database field values. The JOB-ID column specifies format code FM, which defines the column as text rather than as numeric values. The SALARY column, however, is defined as a 6-byte packed decimal numeric field and the NUMBER-OF-EMPLOYEES column is defined as a 4-byte zoned decimal field.

- Report 02 defines four work fields. Three of these fields, JOB-ID, SALARY, and TITLE receive database field values within the type 7 procedure logic of the report. NUMBER-OF-EMPLOYEES is defined as a numeric work field with a value of 1. The field is specified on both a type 5 and a type 6 edit parameter. When the JOB-ID field changes, CA Culprit outputs the total number of records (employees) processed for that particular control break.

- Report 02 specifies both type 7 and type 8 process parameters. The type 7 process parameters move the database field values into work fields; the type 8 process parameter outputs rows to the table when the job-id value changes on the SORT parameter. The table will not contain a row for the grand total.

Example 3—Generating Reports from a Table

In this example, CA Culprit uses EMPLOYEE TABLE, which was created in Example 1, as input. The output from this program includes a report that prints the contents of EMPLOYEE TABLE and a new table that contains a subset of the information contained in EMPLOYEE TABLE. The following figure lists the CA Culprit parameters coded for the program.

Parameters Coded for Example 3:

```
DATABASE DICTNAME=ASFDICT
PROFILE USER=LHN PW=ROSEBUD
INPUT TABLE='EMPLOYEE TABLE' TYPE=COPY VALIDATE=ALL
01$
01$       OUTPUT TABLE EMPLOYEE TABLE
01$
01OUT D
0151*010 EMP-NAME     HF
0151*020 STREET       HF
0151*030 CITY         HF
0151*040 STATE        HF
0151*050 PHONE        HF
02$
02$      CREATE MELROSE EMPLOYEES FROM EMPLOYEE TABLE
02$
02OUTPUT TABLE='MELROSE EMPLOYEES' TYPE=CREATE ONLINE=YES
0251*010 EMP-NAME   SZ=25
0251*020 STREET     SZ=20
0251*030 PHONE      SZ=12
027010   IF STATE NE 'MA' DROP
027      IF CITY NE 'MELROSE' DROP
```

The DATABASE parameter identifies ASFDICT as the dictionary that defines the table. The PROFILE parameter identifies the id and the password of the user who is accessing the table. The INPUT parameter instructs CA Culprit to access table EMPLOYEE TABLE. Since the INPUT parameter does not specify the owner of the table or the dictionary in which the table is defined, CA Culprit defaults to the user ID coded on the PROFILE parameter and the dictionary specified on the DATABASE parameter.

Report 01 prints all the columns of EMPLOYEE TABLE except for START-DATE. The type 5 edit parameters specify the column names as field names. Note that column PHONE is output as a formatted

Printed Contents of a Portion of Table EMPLOYEE TABLE:

```
        EMP-NAME                STREET                CITY            STATE          PHONE

ELEANOR   PEOPLES          756 YELLOWSTONE DR      BOSTON              MA         617-329-1212
ALAN      DONOVAN          6781 CORNWALL AVE       MELROSE             MA         617-665-5412
ALBERT    BREEZE           100 BOARDWALK           OCEAN CITY          NJ         617-554-2387
BETH  M.  CLOUD            3456 PINKY LN           NATICK              MA         617-432-1212
BETSY     ZEDI             34 VALE AVE             SOUTHBORO           MA         617-431-9909
BRIAN     NICEMAN          60 FLORENCE AVE         MELROSE             MA         617-665-4315
BURT      LANCHESTER       45 PINKERTON AVE        WALTHAM             MA         617-534-1109
CAROL     MCDOUGALL        19 URITOP DR            WELLESLEY           MA         617-887-1324
CAROLYN   CROW             891 SUMMER ST           WESTWOOD            MA         617-329-1776
CYNTHIA   JOHNSON          17 MANIFESTO DR         WALPOLE             MA         617-777-8888
        -                        -                    -               -              -
        -                        -                    -               -              -
EDWARD    HUTTON           781 CROSS ST            M    ELROSE              MA         617-665-1010
        -                        -                    -               -              -
        -                        -                    -               -              -
RUPERT    JENSON           999 HARVEY ST           MELROSE             MA         617-665-5556
```

Report 02 creates MELROSE EMPLOYEES, a table that contains the name, street address, and phone number of all employees who live in Melrose, MA. CA Culprit establishes LHN as the owner of MELROSE EMPLOYEES, which is defined to ASFDICT. The contents of table MELROSE EMPLOYEES

Contents of Table MELROSE EMPLOYEES:

```
          EMP-NAME                    STREET                    PHONE

ALAN      DONOVAN                6781 CORNWALL A              617-665-5412
BRIAN     NICEMAN                60 FLORENCE AVE              617-665-4315
DORIS     KING                   716 MORRIS ST                617-665-6161
EDWARD    HUTTON                 781 CROSS S     T                 617-665-1010
RUPERT    JENSON                 999 HARVEY ST                617-665-5556
```

Example 4—Consolidating Tables

In this example, CA Culprit consolidates three departmental tables. Two of the tables reside on one IDMS/DC-UCF system and the third table resides on another. The output from this program is a report that prints the contents of the three tables. The following figure lists the CA Culprit parameters coded for the program.

Parameters Coded for Example 4:

```
DATABASE DICTNAME=ASFDICT
PROFILE USER=DOC1 PW=DOC1
INPUT TABLE=ACCOUNTING TYPE=COPY VALIDATE=ALL
INPUT TABLE='PUBLIC RELATIONS' TYPE=CONSOL
INPUT TABLE=PERSONNEL TYPE=CONSOL
010  ID
0151*0050DEPT-NAME-0410   HH 'DEPARMENT NAME'
0151*010 EMP-ID-0415      HH 'EMPLOYEE ID'
0151*020 EMP-NAME-0415    HH 'EMPLOYEE NAME'
0152*010 EMP-ID-0415
0152*020 EMP-NAME-0415
017      IF TABLE-ID EQ ID   100
017      MOVE TABLE-ID TO ID
017      TAKE 1
017100   TAKE 2
```

The DATABASE parameter identifies ASFDICT as the default dictionary for the CA Culprit run. The PROFILE parameter identifies the id and the password of the user who is accessing the tables. The three INPUT parameters name the following departmental tables:

- Table ACCOUNTING AND PAYROLL is the primary table. Each column of tables PUBLIC RELATIONS and PERSONNEL is compared to the corresponding column definition in table ACCOUNTING AND PAYROLL.
- Table PUBLIC RELATIONS is a secondary table.
- Table PERSONNEL is a secondary table that resides in a different IDMS/DC-UCF system than tables ACCOUNTING and PUBLIC RELATIONS. The INPUT parameter identifies the dictionary that defines the table and the ddname of the SYSCTL file that accesses SYSTEM12.

Report 01 prints the contents of each table. The type 7 process logic uses CA Culprit's reserved word TABLE-ID to print only the first occurrence of each department name.

Printed Contents of Three Consolidated Tables:

DEPARMENT NAME	EMPLOYEE ID	EMPLOYEE NAME
ACCOUNTING AND PAYROLL	0069	JUNE BLOOMER
	0100	EDWARD HUTTON
	0011	RUPERT JENSON
	0067	MARIANNE KIMBALL
	0106	DORIS KING
	0101	BRIAN NICEMAN
PUBLIC RELATIONS	0120	MICHAEL ANGELO
	0007	MONTE BANK
	0158	JOCK JACKSON
	9999	CARRIE BALLINGER
	0127	CAROL MCDOUGALL
	0149	LAURA PENMAN
	0476	BETSY ZEDI
PERSONNEL	0081	TOM FITZHUGH
	8683	JUNIPER HEDGEHOG
	0051	CYNTHIA JOHNSON
	0091	MADELINE ORGRATZI
	0013	ELEANOR PEOPLES

Appendix K: Employee Database Subschema

This section contains the following topics:

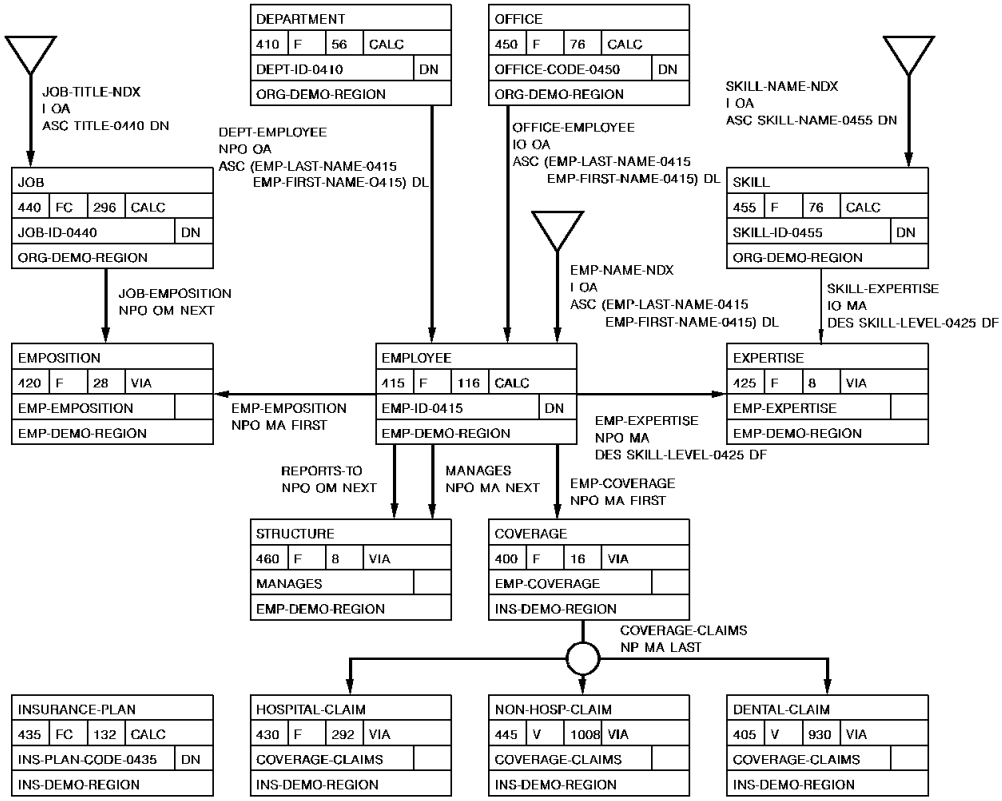
About This Appendix

The employee database is the input for the examples and programs in this manual that reference database records or logical records.

The remainder of this appendix lists the fields defined for each record in the database.

Data Structure Diagram

The following figure shows the structure of the database.



Subschema Listing

Following is a listing of the Employee database:

```
                                SCHEMA=EMPSCHM VERSION=(100)

                                SUBSCHEMA=(EMPSS01)



RECORD NAME........ COVERAGE                                        RLGTH=   36
RECORD VERSION..... 0100                                            DLGTH=   20
RECORD ID.......... 0400                                            KLGTH=   16
RECORD LENGTH...... FIXED                                           DSTRT=   16
LOCATION MODE...... VIA SET      EMP-COVERAGE        DISPLACEMENT 0000 PAGES
WITHIN............. INS-DEMO-REGION     OFFSET       5 PGS FOR        20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    EMP-COVERAGE     MEMBER          1     2     3
                    COVERAGE-CLAIMS  OWNER           4

DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 SELECTION-DATE-0400           DISPLAY                          1     8
03 SELECTION-YEAR-0400           DISPLAY              9(4)        1     4
03 SELECTION-MONTH-0400          DISPLAY              9(2)        5     2
03 SELECTION-DAY-0400            DISPLAY              9(2)        7     2
02 TERMINATION-DATE-0400         DISPLAY                          9     8
03 TERMINATION-YEAR-0400         DISPLAY              9(4)        9     4
03 TERMINATION-MONTH-0400        DISPLAY              9(2)       13     2
03 TERMINATION-DAY-0400          DISPLAY              9(2)       15     2
02 TYPE-0400                     DISPLAY              X          17     1
88 MASTER-0400                   COND         'M'                17
88 FAMILY-0400                   COND         'F'                17
88 DEPENDENT-0400                COND         'D'                17
02 INS-PLAN-CODE-0400            DISPLAY              X(3)       18     3
88 GROUP-LIFE-0400               COND         '001'              18
88 HMO-0400                      COND         '002'              18
88 GROUP-HEALTH-0400             COND         '003'              18
88 GROUP-DENTAL-0400             COND         '004'              18
```

```
RECORD NAME........ DENTAL-CLAIM                                  RLGTH=  944
RECORD VERSION..... 0100                                          DLGTH=  936
RECORD ID.......... 0405                                          KLGTH=    8
RECORD LENGTH...... VARIABLE                                      DSTRT=   12
MINIMUM ROOT.......   132  CHARACTERS
MINIMUM FRAGMENT...   932  CHARACTERS
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS     DISPLACEMENT 0000 PAGES
WITHIN............. INS-DEMO-REGION     OFFSE        5 PGS FOR        20 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
                    (FRAGMENT CHAIN) INTRNL          2

DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0405               DISPLAY                          1     8
03 CLAIM-YEAR-0405               DISPLAY                9(4)      1     4
03 CLAIM-MONTH-0405              DISPLAY                9(2)      5     2
03 CLAIM-DAY-0405                DISPLAY                9(2)      7     2
02 PATIENT-NAME-0405             DISPLAY                          9    25
03 PATIENT-FIRST-NAME-0405       DISPLAY                X(10)     9    10
03 PATIENT-LAST-NAME-0405        DISPLAY                X(15)    19    15
02 PATIENT-BIRTH-DATE-0405       DISPLAY                          34     8
03 PATIENT-BIRTH-YEAR-0405       DISPLAY                9(4)      34     4
03 PATIENT-BIRTH-MONTH-0405      DISPLAY                9(2)      38     2
03 PATIENT-BIRTH-DAY-0405        DISPLAY                9(2)      40     2

02 PATIENT-SEX-0405              DISPLAY                X         42     1
02 RELATION-TO-EMPLOYEE-0405     DISPLAY                X(10)     43    10
02 DENTIST-NAME-0405             DISPLAY                          53    25
03 DENTIST-FIRST-NAME-0405       DISPLAY                X(10)     53    10
03 DENTIST-LAST-NAME-0405        DISPLAY                X(15)     63    15
02 DENTIST-ADDRESS-0405          DISPLAY                          78    46
03 DENTIST-STREET-0405           DISPLAY                X(20)     78    20
03 DENTIST-CITY-0405             DISPLAY                X(15)     98    15
03 DENTIST-STATE-0405            DISPLAY                X(2)     113     2
03 DENTIST-ZIP-0405              DISPLAY                         115     9
04 DENTIST-ZIP-FIRST-FIVE-0405   DISPLAY                X(5)     115     5
04 DENTIST-ZIP-LAST-FOUR-0405    DISPLAY                X(4)     120     4
02 DENTIST-LICENSE-NUMBER-0405   DISPLAY                9(6)     124     6
02 NUMBER-OF-PROCEDURES-0405     COMP                   9(2)     130     2
02 FILLER                        DISPLAY                X        132     1
02 DENTIST-CHARGES-0405          DISPLAY  OCCURS 0 TO 10         133   800
   DEPENDING ON --- NUMBER-OF-PROCEDURES-0405
```

```
03 TOOTH-NUMBER-0405             DISPLAY                9(2)        1     2
03 SERVICE-DATE-0405             DISPLAY                            3     8
04 SERVICE-YEAR-0405             DISPLAY                9(4)        3     4
04 SERVICE-MONTH-0405            DISPLAY                9(2)        7     2
04 SERVICE-DAY-0405              DISPLAY                9(2)        9     2
03 PROCEDURE-CODE-0405           DISPLAY                9(4)       11     4
03 DESCRIPTION-OF-SERVICE-0405   DISPLAY                X(60)      15    60
03 FEE-0405                      COMP-3                 S9(7)V99   75     5
03 FILLER                        DISPLAY                X          80     1


RECORD NAME........ DEPARTMENT                                    RLGTH=   72
RECORD VERSION..... 0100                                          DLGTH=   56
RECORD ID.......... 0410                                          KLGTH=   16
RECORD LENGTH...... FIXED                                         DSTRT=   16
LOCATION MODE...... CALC USING   DEPT-ID-0410         DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC            MEMBER          1     2
                    DEPT-EMPLOYEE   INDEX OWNER     3     4

DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 DEPT-ID-0410                  DISPLAY                9(4)        1     4
02 DEPT-NAME-0410                DISPLAY                X(45)       5    45
02 DEPT-HEAD-ID-0410             DISPLAY                9(4)       50     4
02 FILLER                        DISPLAY                XXX        54     3


RECORD NAME........ EMPLOYEE                                      RLGTH=  192
RECORD VERSION..... 0100                                          DLGTH=  120
RECORD ID.......... 0415                                          KLGTH=   72
RECORD LENGTH...... FIXED                                         DSTRT=   72
LOCATION MODE...... CALC USING   EMP-ID-0415          DUPLICATES NOT ALLOWED
WITHIN............. EMP-DEMO-REGION     OFFSET        5 PGS FOR       45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC            MEMBER          1     2
                    DEPT-EMPLOYEE   INDEX MEMBER    3           4
                    EMP-NAME-NDX    INDEX MEMBER    5
                    EMP-SSN-NDX     INDEX MEMBER    6
                    OFFICE-EMPLOYEE INDEX MEMBER    7           8
                    EMP-COVERAGE    OWNER           9    10
                    EMP-EMPOSITION  OWNER          11    12
                    EMP-EXPERTISE   OWNER          13    14
                    MANAGES         OWNER          15    16
                    REPORTS-TO      OWNER          17    18
```

```
DATA ITEM..... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 EMP-ID-0415              DISPLAY                   9(4)       1     4
02 EMP-NAME-0415            DISPLAY                              5    25
03 EMP-FIRST-NAME-0415      DISPLAY                   X(10)      5    10
03 EMP-LAST-NAME-0415       DISPLAY                   X(15)     15    15
02 EMP-ADDRESS-0415         DISPLAY                             30    46
03 EMP-STREET-0415          DISPLAY                   X(20)     30    20
03 EMP-CITY-0415            DISPLAY                   X(15)     50    15
03 EMP-STATE-0415           DISPLAY                   X(2)      65     2
03 EMP-ZIP-0415             DISPLAY                             67     9
04 EMP-ZIP-FIRST-FIVE-0415  DISPLAY                   X(5)      67     5
04 EMP-ZIP-LAST-FOUR-0415   DISPLAY                   X(4)      72     4
02 EMP-PHONE-0415           DISPLAY                   9(10)     76    10
02 STATUS-0415              DISPLAY                   X(2)      86     2
88 ACTIVE-0415              COND          '01'                  86
88 ST-DISABIL-0415          COND          '02'                  86
88 LT-DISABIL-0415          COND          '03'                  86
88 LEAVE-OF-ABSENCE-0415    COND          '04'                  86
88 TERMINATED-0415          COND          '05'                  86

02 SS-NUMBER-0415           DISPLAY                   9(9)      88     9
02 START-DATE-0415          DISPLAY                             97     8
03 START-YEAR-0415          DISPLAY                   9(4)      97     4
03 START-MONTH-0415         DISPLAY                   9(2)     101     2
03 START-DAY-0415           DISPLAY                   9(2)     103     2
02 TERMINATION-DATE-0415    DISPLAY                            105     8
03 TERMINATION-YEAR-0415    DISPLAY                   9(4)     105     4
03 TERMINATION-MONTH-0415   DISPLAY                   9(2)     109     2
03 TERMINATION-DAY-0415     DISPLAY                   9(2)     111     2
02 BIRTH-DATE-0415          DISPLAY                            113     8
03 BIRTH-YEAR-0415          DISPLAY                   9(4)     113     4
03 BIRTH-MONTH-0415         DISPLAY                   9(2)     117     2
03 BIRTH-DAY-0415           DISPLAY                   9(2)     119     2


RECORD NAME........ EMPOSITION                                RLGTH=   56
RECORD VERSION..... 0100                                      DLGTH=   32
RECORD ID.......... 0420                                      KLGTH=   24
RECORD LENGTH...... FIXED                                     DSTRT=   24
LOCATION MODE...... VIA SET      EMP-EMPOSITION     DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    EMP-EMPOSITION   MEMBER          1     2     3
                    JOB-EMPOSITION   MEMBER          4     5     6
```

```
DATA ITEM..... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 START-DATE-0420          DISPLAY                          1     8
03 START-YEAR-0420          DISPLAY              9(4)        1     4
03 START-MONTH-0420         DISPLAY              9(2)        5     2
03 START-DAY-0420           DISPLAY              9(2)        7     2
02 FINISH-DATE-0420         DISPLAY                          9     8
03 FINISH-YEAR-0420         DISPLAY              9(4)        9     4
03 FINISH-MONTH-0420        DISPLAY              9(2)       13     2
03 FINISH-DAY-0420          DISPLAY              9(2)       15     2
02 SALARY-GRADE-0420        DISPLAY              9(2)       17     2
02 SALARY-AMOUNT-0420       COMP-3               S9(7)V99   19     5
02 BONUS-PERCENT-0420       COMP-3               SV999      24     2
02 COMMISSION-PERCENT-0420  COMP-3               SV999      26     2
02 OVERTIME-RATE-0420       COMP-3               S9V99      28     2
02 FILLER                   DISPLAY              XXX        30     3


RECORD NAME........ EXPERTISE                                RLGTH=   32
RECORD VERSION..... 0100                                     DLGTH=   12
RECORD ID.......... 0425                                     KLGTH=   20
RECORD LENGTH...... FIXED                                    DSTRT=   20
LOCATION MODE...... VIA SET      EMP-EXPERTIS       DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    EMP-EXPERTISE    MEMBER          1     2     3
                    SKILL-EXPERTISE  INDEX MEMBER    4           5
DATA ITEM..... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH

02 SKILL-LEVEL-0425         DISPLAY                XX         1     2
88 EXPERT-0425              COND         '04'                  1
88 PROFICIENT-0425          COND         '03'                  1
88 COMPETENT-0425           COND         '02'                  1
88 ELEMENTARY-0425          COND         '01'                  1
02 EXPERTISE-DATE-0425      DISPLAY                            3     8
03 EXPERTISE-YEAR-0425      DISPLAY                9(4)        3     4
03 EXPERTISE-MONTH-0425     DISPLAY                9(2)        7     2
03 EXPERTISE-DAY-0425       DISPLAY                9(2)        9     2
02 FILLER                   DISPLAY                XX         11     2
```

```
RECORD NAME........ HOSPITAL-CLAIM                                RLGTH=  304
RECORD VERSION..... 0100                                          DLGTH=  300
RECORD ID.......... 0430                                          KLGTH=    4
RECORD LENGTH...... FIXED                                         DSTRT=    4
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS    DISPLACEMENT 0000  PAGES
WITHIN............. INS-DEMO-REGION     OFFSET      5 PGS FOR         20 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0430                DISPLAY                          1     8
03 CLAIM-YEAR-0430                DISPLAY                 9(4)     1     4
03 CLAIM-MONTH-0430               DISPLAY                 9(2)     5     2
03 CLAIM-DAY-0430                 DISPLAY                 9(2)     7     2
02 PATIENT-NAME-0430              DISPLAY                          9    25
03 PATIENT-FIRST-NAME-0430        DISPLAY                 X(10)    9    10
03 PATIENT-LAST-NAME-0430         DISPLAY                 X(15)   19    15

02 PATIENT-BIRTH-DATE-0430        DISPLAY                          34     8
03 PATIENT-BIRTH-YEAR-0430        DISPLAY                 9(4)     34     4
03 PATIENT-BIRTH-MONTH-0430       DISPLAY                 9(2)     38     2
03 PATIENT-BIRTH-DAY-0430         DISPLAY                 9(2)     40     2
02 PATIENT-SEX-0430               DISPLAY                 X        42     1
02 RELATION-TO-EMPLOYEE-0430      DISPLAY                 X(10)    43    10
02 HOSPITAL-NAME-0430             DISPLAY                 X(25)    53    25
02 HOSP-ADDRESS-0430              DISPLAY                          78    46
03 HOSP-STREET-0430               DISPLAY                 X(20)    78    20
03 HOSP-CITY-0430                 DISPLAY                 X(15)    98    15
03 HOSP-STATE-0430                DISPLAY                 X(2)    113     2
03 HOSP-ZIP-0430                  DISPLAY                         115     9
04 HOSP-ZIP-FIRST-FIVE-0430       DISPLAY                 X(5)    115     5

04 HOSP-ZIP-LAST-FOUR-0430        DISPLAY                 X(4)    120     4
02 ADMIT-DATE-0430                DISPLAY                         124     8
03 ADMIT-YEAR-0430                DISPLAY                 9(4)    124     4
03 ADMIT-MONTH-0430               DISPLAY                 9(2)    128     2
03 ADMIT-DAY-0430                 DISPLAY                 9(2)    130     2
02 DISCHARGE-DATE-0430            DISPLAY                         132     8
03 DISCHARGE-YEAR-0430            DISPLAY                 9(4)    132     4
03 DISCHARGE-MONTH-0430           DISPLAY                 9(2)    136     2
03 DISCHARGE-DAY-0430             DISPLAY                 9(2)    138     2
02 DIAGNOSIS-0430                 DISPLAY  OCCURS 2       X(60)   140   120
02 HOSPITAL-CHARGES-0430          DISPLAY                         260    41
03 ROOM-AND-BOARD-0430            DISPLAY                         260    26
04 WARD-0430                      DISPLAY                         260    13
05 WARD-DAYS-0430                 COMP-3                  S9(5)   260     3
05 WARD-RATE-0430                 COMP-3                  S9(7)V99 263    5
05 WARD-TOTAL-0430                COMP-3                  S9(7)V99 268    5
04 SEMI-PRIVATE-0430              DISPLAY                         273    13
```

```
05 SEMI-DAYS-0430                COMP-3                  S9(5)     273     3
05 SEMI-RATE-0430                COMP-3                  S9(7)V99  276     5
05 SEMI-TOTAL-0430               COMP-3                  S9(7)V99  281     5
03 OTHER-CHARGES-0430            DISPLAY                           286    15
04 DELIVERY-COST-0430            COMP-3                  S9(7)V99  286     5
04 ANESTHESIA-COST-0430          COMP-3                  S9(7)V99  291     5
04 LAB-COST-0430                 COMP-3                  S9(7)V99  296     5


RECORD NAME........ INSURANCE-PLAN                                RLGTH=  140
RECORD VERSION..... 0100                                          DLGTH=  132
RECORD ID.......... 0435                                          KLGTH=    8
RECORD LENGTH...... FIXED                                         DSTRT=    8
LOCATION MODE...... CALC USING   INS-PLAN-CODE-0435   DUPLICATES NOT ALLOWED
WITHIN............. INS-DEMO-REGION     OFFSET        1 PGS FOR        4 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC            MEMBER          1     2
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 INS-PLAN-CODE-0435            DISPLAY                 X(3)        1     3

88 GROUP-LIFE-0435               COND         '001'                  1
88 HMO-0435                      COND         '002'                  1
88 GROUP-HEALTH-0435             COND         '003'                  1
88 GROUP-DENTAL-0435             COND         '004'                  1

02 INS-CO-NAME-0435              DISPLAY                 X(45)       4    45
02 INS-CO-ADDRESS-0435           DISPLAY                             49    46
03 INS-CO-STREET-0435            DISPLAY                 X(20)       49    20
03 INS-CO-CITY-0435              DISPLAY                 X(15)       69    15
03 INS-CO-STATE-0435             DISPLAY                 X(2)        84     2
03 INS-CO-ZIP-0435               DISPLAY                             86     9
04 INS-CO-ZIP-FIRST-FIVE-0435    DISPLAY                 X(5)        86     5
04 INS-CO-ZIP-LAST-FOUR-0435     DISPLAY                 X(4)        91     4
02 INS-CO-PHONE-0435             DISPLAY                 9(10)       95    10
02 GROUP-NUMBER-0435             DISPLAY                 9(6)       105     6
02 PLAN-DESCRIPTION-0435         DISPLAY                            111    20
03 DEDUCT-0435                   COMP-3                  S9(7)V99   111     5
03 MAXIMUM-LIFE-COST-0435        COMP-3                  S9(7)V99   116     5
03 FAMILY-COST-0435              COMP-3                  S9(7)V99   121     5
03 DEP-COST-0435                 COMP-3                  S9(7)V99   126     5
02 FILLER                        DISPLAY                 XX         131     2
```

```
RECORD NAME........ JOB                                          RLGTH=  324
RECORD VERSION..... 0100                                         DLGTH=  300
RECORD ID.......... 0440                                         KLGTH=   24
RECORD LENGTH...... FIXED  (INTERNALLY VARIABLE)                 DSTRT=   28
MINIMUM ROOT.......    24  CHARACTERS
MINIMUM FRAGMENT...   296  CHARACTERS
LOCATION MODE...... CALC USING   JOB-ID-0440          DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
CALL PROCEDURES.... NAME.... WHEN.. FUNCTION
                    IDMSCOMP BEFORE STORE
                    IDMSCOMP BEFORE MODIFY
                    IDMSDCOM AFTER  GET
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    JOB-TITLE-NDX    INDEX MEMBER    3
                    JOB-EMPOSITION   OWNER           4     5
                    (FRAGMENT CHAIN) INTRNL          6

DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 JOB-ID-0440                   DISPLAY                 9(4)        1     4
02 TITLE-0440                    DISPLAY                 X(20)       5    20

02 DESCRIPTION-0440              DISPLAY                              25   120
03 DESCRIPTION-LINE-0440         DISPLAY  OCCURS 2       X(60)       25   120
02 REQUIREMENTS-0440             DISPLAY                             145   120
03 REQUIREMENT-LINE-0440         DISPLAY  OCCURS 2       X(60)      145   120
02 MINIMUM-SALARY-0440           DISPLAY                 S9(6)V99   265     8
02 MAXIMUM-SALARY-0440           DISPLAY                 S9(6)V99   273     8
02 SALARY-GRADES-0440            DISPLAY  OCCURS 4       9(2)       281     8
02 NUMBER-OF-POSITIONS-0440      DISPLAY                 9(3)       289     3
02 NUMBER-OPEN-0440              DISPLAY                 9(3)       292     3
02 FILLER                        DISPLAY                 XX         295     2

RECORD NAME........ NON-HOSP-CLAIM                               RLGTH= 1064
RECORD VERSION..... 0100                                         DLGTH= 1056
RECORD ID.......... 0445                                         KLGTH=    8
RECORD LENGTH...... VARIABLE                                     DSTRT=   12
MINIMUM ROOT.......   248  CHARACTERS
MINIMUM FRAGMENT...  1052  CHARACTERS
LOCATION MODE...... VIA SET      COVERAGE-CLAIMS    DISPLACEMENT 0000  PAGES
WITHIN............. INS-DEMO-REGION     OFFSET      5 PGS FOR         20 PGS
DBKEY POSITIONS.... SET............. TYPE........ NEXT  PRIOR OWNER
                    COVERAGE-CLAIMS  MEMBER          1
                    (FRAGMENT CHAIN) INTRNL          2
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 CLAIM-DATE-0445               DISPLAY                             1     8
03 CLAIM-YEAR-0445               DISPLAY                 9(4)        1     4
03 CLAIM-MONTH-0445              DISPLAY                 9(2)        5     2
03 CLAIM-DAY-0445                DISPLAY                 9(2)        7     2
02 PATIENT-NAME-0445             DISPLAY                             9    25
```

```
03 PATIENT-FIRST-NAME-0445       DISPLAY                X(10)        9    10
03 PATIENT-LAST-NAME-0445        DISPLAY                X(15)       19    15
02 PATIENT-BIRTH-DATE-0445       DISPLAY                            34     8
03 PATIENT-BIRTH-YEAR-0445       DISPLAY                9(4)        34     4
03 PATIENT-BIRTH-MONTH-0445      DISPLAY                9(2)        38     2
03 PATIENT-BIRTH-DAY-0445        DISPLAY                9(2)        40     2
02 PATIENT-SEX-0445              DISPLAY                X           42     1
02 RELATION-TO-EMPLOYEE-0445     DISPLAY                X(10)       43    10
02 PHYSICIAN-NAME-0445           DISPLAY                            53    25

03 PHYSICIAN-FIRST-NAME-0445     DISPLAY                X(10)       53    10
03 PHYSICIAN-LAST-NAME-0445      DISPLAY                X(15)       63    15

02 PHYSICIAN-ADDRESS-0445        DISPLAY                            78    46
03 PHYSICIAN-STREET-0445         DISPLAY                X(20)       78    20
03 PHYSICIAN-CITY-0445           DISPLAY                X(15)       98    15
03 PHYSICIAN-STATE-0445          DISPLAY                X(2)       113     2
03 PHYSICIAN-ZIP-0445            DISPLAY                           115     9
04 PHYSICIAN-ZIP-FIRST-FIVE-0445 DISPLAY                X(5)       115     5
04 PHYSICIAN-ZIP-LAST-FOUR-0445  DISPLAY                X(4)       120     4
02 PHYSICIAN-ID-0445             DISPLAY                9(6)       124     6
02 DIAGNOSIS-0445                DISPLAY  OCCURS 2      X(60)      130   120
02 NUMBER-OF-PROCEDURES-0445     COMP                   9(2)       250     2
02 FILLER                        DISPLAY                X          252     1
02 PHYSICIAN-CHARGES-0445        DISPLAY  OCCURS 0 TO 10           253   800
   DEPENDING ON -- NUMBER-OF-PROCEDURES-0445

03 SERVICE-DATE-0445             DISPLAY                             1     8
04 SERVICE-YEAR-0445             DISPLAY                9(4)         1     4
04 SERVICE-MONTH-0445            DISPLAY                9(2)         5     2
04 SERVICE-DAY-0445              DISPLAY                9(2)         7     2
03 PROCEDURE-CODE-0445           DISPLAY                9(4)         9     4
03 DESCRIPTION-OF-SERVICE-0445   DISPLAY                X(60)       13    60
03 FEE-0445                      COMP-3                 S9(7)V99    73     5
03 FILLER                        DISPLAY                XXX         78     3
```

```
RECORD NAME........ OFFICE                                        RLGTH=   92
RECORD VERSION..... 0100                                          DLGTH=   76
RECORD ID.......... 0450                                          KLGTH=   16
RECORD LENGTH...... FIXED                                         DSTRT=   16
LOCATION MODE...... CALC USING   OFFICE-CODE-0450     DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    OFFICE-EMPLOYEE  INDEX OWNER     3     4
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 OFFICE-CODE-0450              DISPLAY                 X(3)        1     3
02 OFFICE-ADDRESS-0450           DISPLAY                             4    46
03 OFFICE-STREET-0450            DISPLAY                 X(20)       4    20
03 OFFICE-CITY-0450              DISPLAY                 X(15)      24    15

03 OFFICE-STATE-0450             DISPLAY                 X(2)       39     2
03 OFFICE-ZIP-0450               DISPLAY                             41     9
04 OFFICE-ZIP-FIRST-FIVE-0450    DISPLAY                 X(5)       41     5
04 OFFICE-ZIP-LAST-FOUR-0450     DISPLAY                 X(4)       46     4
02 OFFICE-PHONE-0450             DISPLAY  OCCURS 3       9(7)       50    21
02 OFFICE-AREA-CODE-0450         DISPLAY                 X(3)       71     3

02 SPEED-DIAL-0450               DISPLAY                 X(3)       74     3

RECORD NAME........ SKILL                                         RLGTH=   96
RECORD VERSION..... 0100                                          DLGTH=   76
RECORD ID.......... 0455                                          KLGTH=   20
RECORD LENGTH...... FIXED                                         DSTRT=   20

LOCATION MODE...... CALC USING   SKILL-ID-0455        DUPLICATES NOT ALLOWED
WITHIN............. ORG-DEMO-REGION     OFFSET        5 PGS FOR       20 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    CALC             MEMBER          1     2
                    SKILL-NAME-NDX   INDEX MEMBER    3
                    SKILL-EXPERTISE  INDEX OWNER     4     5
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
02 SKILL-ID-0455                 DISPLAY                 9(4)        1     4
02 SKILL-NAME-0455               DISPLAY                 X(12)       5    12
02 SKILL-DESCRIPTION-0455        DISPLAY                 X(60)      17    60

RECORD NAME........ STRUCTURE                                     RLGTH=   36
RECORD VERSION..... 0100                                          DLGTH=   12
RECORD ID.......... 0460                                          KLGTH=   24
RECORD LENGTH...... FIXED                                         DSTRT=   24
LOCATION MODE...... VIA SET      MANAGES            DISPLACEMENT 0000  PAGES
WITHIN............. EMP-DEMO-REGION     OFFSET      5 PGS FOR         45 PGS
DBKEY POSITIONS.... SET.............. TYPE........ NEXT  PRIOR OWNER
                    MANAGES          MEMBER          1     2     3
                    REPORTS-TO       MEMBER          4     5     6
DATA ITEM.......... REDEFINES... USAGE....... VALUE..... PICTURE. STRT  LGTH
```

```
02 STRUCTURE-CODE-0460          DISPLAY                X(2)       1     2
88 ADMIN-0460                   COND 'A'                          1
88 PROJECT-0460                 COND 'P1' THRU 'P9'               1
02 STRUCTURE-DATE-0460          DISPLAY                           3     8
03 STRUCTURE-YEAR-0460          DISPLAY                9(4)       3     4
03 STRUCTURE-MONTH-0460         DISPLAY                9(2)       7     2
03 STRUCTURE-DAY-0460           DISPLAY                9(2)       9     2
02 FILLER                       DISPLAY                XX        11     2
```

Appendix L: The DB-EXIT Facility

This section contains the following topics:

About DB-EXIT

The DB-EXIT facility accesses database records or logical records directly within type 7 process logic. The DB-EXIT facility is invoked in type 7 process logic with either a CALL DB-EXIT statement or a series of MOVE statements followed by a branch to DB-EXIT, as follows:

```
017010 CALL DB-EXIT ('FIRST','EMPLOYEE ','DEPT-EMPLOYEE ')

              or

017010 MOVE 'FIRST'          TO      ARG1
017    MOVE 'EMPLOYEE '      TO      ARG2
017    MOVE 'DEPT-EMPLOYEE ' TO      ARG3
017    B     DB-EXIT
```

Usage Notes

Where to Code It

Records that are to be retrieved by using the DB-EXIT facility appear on dummy PATH parameters. Dummy PATH parameters specify a primary path id of two hyphens (--). The following considerations apply to dummy PATH parameters:

- Records on dummy paths do not have to participate in set relationships with one another because the user controls record retrieval.
- Dummy paths can be the only paths included in the database access run. In such cases, no values are set in the input buffer the first time CA Culprit enters the report's procedure logic. When CA Culprit encounters a DB-EXIT call, it retrieves records from the database and fills areas of the input buffer that are reserved on the dummy paths. After each report in the run has issued a DROP or TAKE command, CA Culprit reenters the procedure logic of the first report.

  Note: A STOP-RUN command must be coded on a type 7 parameter to terminate a CA Culprit run that specifies only a dummy path.

- Each dummy path parameter can specify a different logical record; each logical record can specify a WHERE clause. CA Culprit evaluates the WHERE clause when it executes the DB-EXIT call.

Automatic Record Retrieval

The DB-EXIT facility can also be used to retrieve records that appear on automatic retrieval paths. When this occurs, currency for the automatic retrieval path can be affected as follows:

- If the DB-EXIT call references the record level corresponding to a record on the dummy path, currencies on automatic retrieval paths are not affected.
- If the DB-EXIT call references the record level corresponding to a record on an automatic retrieval path, currencies on the automatic retrieval path can be changed. Care must be applied in such cases.

Each DB-EXIT call requires from one to six argument values. This table lists the arguments required for each particular DB-EXIT call.

Values of Arguments Associated with DB-EXIT:

ARG1	ARG2	ARG3	ARG4	ARG5	ARG6
DBKEY*	database-record-name(.level-n)*#	dbkey-value-x	--	--	--
FORMAT-6*	database-record-name(.level-n)*#	set-name*#	key-field-name	key-value-qvnx	key-length-vn
CALC*	database-record-name(.level-n)*#	calc-key-field-name	calc-key-value-qvnx	calc-key-length-vn	--
DUP*	database-record-name(.level-n)*#	calc-key-field-name	calc-key-value-qvnx	calc-key-length-vn	--
OWNER*	database-record-name(.level-n)*#	set-name*#	--	--	--
NEXT*	database-record-name(.level-n)*#	set-name*#	--	--	--
PRIOR*	database-record-name(.level-n)*#	set-name*#	--	--	--
FIRST*	database-record-name(.level-n)*#	set-name*#	--	--	--
LAST*	database-record-name(.level-n)*#	set-name*#	--	--	--
NTH*	database-record-name(.level-n)*#	set-name*#	occurrence-vn	--	--
NEXT-AREA*	database-record-name(.level-n)*#	area-name*#	--	--	--
PRIOR-AREA*	database-record-name(.level-n)*#	area-name*#	--	--	--
FIRST-AREA*	database-record-name(.level-n)*#	area-name*#	--	--	--
LAST-AREA*	database-record-name(.level-n)*#	area-name*#	--	--	--
NTH-AREA*	database-record-name(.level-n)*#	area-name*#	occurrence-vn	--	--
IF-MEMBER*	database-record-name(.level-n)*#	set-name*#	--	--	--
SET-VALUE*	field-name	field-value-qvnx	field-length-vn	field-data-type-qv	--
LR-FIRST*	logical-record-name(.level-n)*#	--	--	--	--
LR-NEXT*	logical-record-name(.level-n)*#	--	--	--	--

Notes:

* Code a literal value in quotes or refer to an alphanumeric field equal to the literal.

Literal values must end in a space.

The following suffix conventions are used for variable items in the above table. A suffix comprises all entries appearing after the last hyphen in a variable item; multiple entries indicate acceptable alternatives.

Notation	Meaning
-a	User-supplied value
-n	Numeric literal
-q	Value enclosed in quotation marks
-v	Variable data field containing an appropriate value or targeted to receive data
-x	Hexadecimal literal denoted as X'*variable*'

When CA Culprit processes a DB-EXIT call, it sets values in the reserved input buffer fields, IDMS-STATUS and LR-STATUS, as follows:

- Values assigned to the IDMS-STATUS field differ under the following circumstances:
  - If one or more arguments listed for a call is invalid, CA Culprit sets IDMS-STATUS to *00*n*, where *n* is the number of the first argument in which an error was detected.
  - If all the arguments are valid and CA Culprit retrieves database records using the DB-EXIT facility, the IDMS-STATUS field is set to the status code returned by CA IDMS/DB. The following table defines the status codes most frequently returned.

    Note: For more information about error status codes, see the *CA IDMS Messages and Codes Guide*.

Code	Description
0000	No run-time errors occurred.
0326	Record was not found by using a CALC key, sort key, or index key.
0307	Record was not returned due to an end-of-set (or an end-of-area) condition, which results from an attempt to find the next or prior record in the set (or area).

- Values assigned to the LR-STATUS field are set when the dummy path specifies a logical record. The field is set to the logical record status returned to CA Culprit by CA IDMS/DB.

Note: It is the user's responsibility to check IDMS-STATUS and LR-STATUS after each DB-EXIT call and to specify appropriate action in the procedure logic, as shown in the following example:

```
017010 IDMS-STATUS EQ '0307' DROP
```

Each DB-EXIT argument is described below:

ARG1

ARG1 is an alphanumeric literal or work field that determines the type of CA IDMS/DB retrieval command issued. The following table lists the possible values for ARG1 and associated commands, except for SET-VALUE.

When CA Culprit encounters SET-VALUE, it does not access the database. SET-VALUE sets a value in an input buffer field. This value is generally referenced in a WHERE clause for a logical record obtained in a subsequent LR-FIRST or LR-NEXT DB-EXIT call.

If SET-VALUE is used to set a value in a logical record, the SET-VALUE call must be performed before each LR-FIRST or LR-NEXT call. Otherwise, the DBA-specified retrieval path may cause the value to be overlayed.

ARG1	Retrieval Command Generated
DBKEY	OBTAIN *database-record-name* DBKEY IS *dbkey-field-name*
FORMAT-6	OBTAIN *database-record-name* WITHIN *ix-set-name* USING *key-value-va*
CALC	OBTAIN CALC *database-record-name*
DUP	OBTAIN DUP *database-record-name* (This command must be preceded by a CALC DB-EXIT call.)
OWNER	OBTAIN OWNER WITHIN *set-name* (*Database-record-name* in ARG2 is the owner record.)
NEXT	OBTAIN NEXT *database-record-name* WITHIN *set-name*
PRIOR	OBTAIN PRIOR *database-record-name* WITHIN *set-name*
FIRST	OBTAIN FIRST *database-record-name* WITHIN *set-name*
LAST	OBTAIN LAST *database-record-name* WITHIN *set-name*
NTH	OBTAIN *occurrence-vn database-record-name* WITHIN *set-name*
NEXT-AREA	OBTAIN NEXT *database-record-name* WITHIN *area-name*

ARG1	Retrieval Command Generated
PRIOR-AREA	OBTAIN PRIOR *database-record-name* WITHIN *area-name*
FIRST-AREA	OBTAIN FIRST *database-record-name* WITHIN *area-name*
LAST-AREA	OBTAIN LAST *database-record-name* WITHIN *area-name*
NTH-AREA	OBTAIN *occurrence-vn database-record-name* WITHIN *area-name*
IF-MEMBER	FIND CURRENT *database-record-name* IF *set-name* MEMBER
LR-FIRST	OBTAIN FIRST *logical-record-name* WHERE *lr-boolean-expression*
LR-NEXT	OBTAIN NEXT *logical-record-name* WHERE *lr-boolean-expression*

ARG2 Values

ARG2 values vary depending on the value of ARG1.

Note: See the explanation of suffix notation conventions above.

- **Database-record-name/logical-record-name level-n** is an alphanumeric literal or work field that specifies the name of the database record or logical record to be obtained or tested. The following coding considerations apply to the record name:
  - The record name must be defined to the subschema; record name synonyms are not allowed.
  - If the record name is less than 16 characters, the name must be terminated by a space or followed immediately by a comma and *level-n*.
  - The record name and record level, if specified, must be enclosed in single quotation marks.

  Level-n is the level number of the database record or logical record to be obtained; the default is 1. If *level-n* is specified, it must be separated from the record name by a comma, with no intervening spaces, and must be terminated by a space. If ARG2 is a logical record, any WHERE clause associated with that record and level number is applied when an occurrence of that record is retrieved.
- **Field-name** is the name of a field in the input buffer into which the value is to be moved by a SET-VALUE call. *Field-name* must be fully qualified if the field appears in more than one record or if multiple levels of a record are coded.

ARG3 Values

ARG3 values vary depending on the value of ARG1.

Note: See the explanation of suffix notation conventions above.

- **Dbkey-value-vx** is either a 4-byte (8-digit) hexadecimal literal, denoted as X'*variable*', or a binary field that contains the db-key of the database record to be obtained.
- **Set-name/area-name** is an alphanumeric literal or work field that contains the set name or area name of the database record to be accessed. The following coding considerations apply to *set-name* and *area-name*:
  - If the name contains less than 16 characters, it must be terminated by a space.
  - The name must be enclosed in single quotation marks.
- **Calc-key-field-name** is the CALC-key field for the database record to be accessed. If the value of ARG2 contains a record-level qualifier, *calc-key-field-name* must be qualified with *record-level-n* also.

  The CALC key can be a concatenated field that contains one or more noncontiguous database record fields; for example, FIELDA, FIELDB, FIELDC. CA Culprit can retrieve records with concatenated CALC keys by using the CALC DB-EXIT command and a CA Culprit module that performs variable-length moves (CULLUS43). CULLUS43 must be executed for each partial CALC key except the last; the module moves a value to each partial key field. The value for ARG3 in a CALC DB-EXIT call is the name of the last partial CALC-key field.

  Note: For more information about procedure module CULLUS43, see the *CA Culprit for CA IDMS User Modules Guide*.
- **Field-value-qvnx** is either an alphanumeric or numeric literal enclosed in single quotation marks or an alphanumeric or numeric work field. CA Culprit places the value of this argument in the input buffer field specified by *field-name* in ARG2 of a SET-VALUE call.

  The following considerations apply depending on whether the field is alphanumeric or numeric:
  - If the field is alphanumeric, *field-value-qvnx* must be the same length as the field identified by *field-name*.
  - If the field is numeric, *field-value-qvnx* must have the same number of decimal positions as the field identified by *field-name*. Numeric fields with no decimal positions can be either 8- or 16-byte numeric work fields.

ARG4 Values

ARG4 values vary depending on the value of ARG1:

- **Key-field-name** is the sort or index key field in the sorted or indexed set that contains the record to be accessed.
- **Calc-key-value-qvnx** is either an alphanumeric or numeric literal enclosed in single quotation marks, an input field, or an alphanumeric work field that contains the value of the CALC key specified for ARG3. *Calc-key-value-qvnx* must be in exactly the same data format as the field in the record; that is, if the database field is a 6-byte packed decimal field, the supplied key value must be also.
- **Occurrence-vn** is either a numeric literal or work field with no decimal places that contains the occurrence of the record to be accessed with the set or area.
- **Field-length-vn** is either a numeric literal or work field with no decimal places that specifies the length of the field identified by *field-name* in a SET-VALUE call. The maximum value depends on *field-data-type-qv*, as described below and shown in the following table.

Specified Data Type		
Maximum Field	**Data Type**	**After Conversion**
' '	Alphanumeric	256
'1'	Binary	4
'2'	Zoned	31
'3'	Packed signed	16

Note: Data types associated with a field specified in a SET-VALUE call must be enclosed in single quotation marks.

ARG5 Values

ARG5 values vary depending on the value of ARG1:

- **Key-value-qvnx** is either an alphanumeric or numeric literal enclosed in single quotation marks, an input field, or a work field that contains the value of the indexed field of the record to be accessed. *Key-value-qvnx* must be in exactly the same data format as the indexed field in the record.
- **Calc-key-length-vn** is a numeric literal or work field that contains the length of the CALC-key field specified for ARG3. This field must not contain or be defined with any decimal places.

- **Field-data-type-qv** is a 1-byte alphanumeric literal or work field that specifies the data type of the field specified by *field-name* in a SET-VALUE call. Values for *field-data-type-qv* and corresponding maximum values for *field-length-vn* are shown in the earlier table.

  CA Culprit converts numeric fields to the length and data type specified for *field-length-vn* and *field-data-type-qv*, respectively. To perform the conversion, the numeric field must have the following characteristics:

  - The field must be either an 8- or 16-byte packed decimal value.
  - The field must contain the same number of decimal positions as defined for *field-name* in ARG2 of a SET-VALUE call.

ARG6

ARG6 is the last argument when ARG1 specifies FORMAT-6. ARG6 is assigned *key-length-vn*, which is a numeric literal or work field that contains the length of the indexed field, as defined in IDD, for the set that contains the record to be accessed.

Examples

Examples of the DB-EXIT facility are shown and described below.

Example 1

```
PATHAA DEPARTMENT
PATH-- EMPLOYEE

017    CALL DB-EXIT ('FIRST' 'EMPLOYEE ' 'DEPT-EMPLOYEE ')
017    RELS
017030 CALL DB-EXIT ('NEXT' 'EMPLOYEE ' 'DEPT-EMPLOYEE ')
017    IF IDMS-STATUS EQ '0307' DROP
017    RELS
017    B 030
```

The first DB-EXIT call in this example retrieves the first EMPLOYEE record that participates in a set relationship with the DEPARTMENT record currently in the input buffer. The second DB-EXIT call retrieves all remaining EMPLOYEE record occurrences in this set relationship until CA Culprit encounters an IDMS-STATUS code that indicates an end-of-set condition.

The first DB-EXIT call generates an OBTAIN FIRST EMPLOYEE WITHIN DEPT-EMPLOYEE command; the second call generates an OBTAIN NEXT EMPLOYEE WITHIN DEPT-EMPLOYEE command. ARG1 specifies the type of call enclosed in single quotation marks. ARG2 specifies the record name EMPLOYEE followed by a single space and enclosed in single quotation marks; ARG3 specifies the set name DEPT-EMPLOYEE followed by a single space and enclosed in single quotation marks.

Example 2

```
PATH-- EMPLOYEE
017    MOVE 'CALC' TO ARG1
017    MOVE 'EMPLOYEE ' TO ARG2
017    MOVE EMP-ID-0415 TO ARG3
017    MOVE '0301' TO ARG4
017    MOVE 4 TO ARG5
017    B    DB-EXIT
017    RELS
017    IDMS-STATUS EQ '0000'  STOP-RUN
```

The only PATH parameter specified for this database access run is a dummy path parameter that specifies the EMPLOYEE record. Since the code does not include an automatic retrieval PATH parameter, the code must include a STOP-RUN command in order to terminate processing for the run.

In this example, the DB-EXIT call executes a series of MOVE commands followed by a branch to DB-EXIT. ARG1 generates an OBTAIN CALC EMPLOYEE command. ARG2 specifies the EMPLOYEE record name followed by a space. ARG3 specifies the name of the CALC-key field for the EMPLOYEE record. ARG4 specifies the value of the EMP-ID-0415 field, which is defined as numeric, enclosed in single quotation marks; ARG5 specifies the length, in bytes, of the EMP-ID-0415 field.

Example 3

```
017100 CALL DB-EXIT ('CALC' 'EMPLOYEE ' EMP-ID-0415 '0301' 4)
017    IDMS-STATUS NE '0000'  DROP
017    RELS
017    MOVE  'DUP' TO ARG1
017200 B DB-EXIT
017    IDMS-STATUS NE '0000'  DROP
017    RELS
017    B 200
```

This segment of code includes two DB-EXIT requests. The first DB-EXIT request returns an EMPLOYEE record occurrence that has a CALC-key value equal to 0301. The second DB-EXIT request issues an OBTAIN DUPLICATE EMPLOYEE command, which returns any other EMPLOYEE records with a CALC-key value equal to 0301.

The argument values for both DB-EXIT requests are the same except for the value of ARG1. A MOVE statement changes the value of ARG1 from CALC to DUP before process statement 200 is executed.

Example 4

```
 PATH-- EMPLOYEE
 010    EMP-ID  '0301'
 010    EMP-SS-NUM  '123456789'

 017010 CALL US43 (EMP-ID,EMP-ID-0415,4)
 017    CALL DB-EXIT ('CALC','EMPLOYEE ',EMP-SS-NUM-0415,
*                     EMP-SS-NUM,13)
```

The CALC key for the EMPLOYEE record consists of two fields: EMP-ID-0415 and EMP-SS-NUM-0415. The call to procedure module US43 moves the value in work field EMP-ID to EMP-ID-0415. The DB-EXIT call generates an OBTAIN CALC EMPLOYEE command. Arguments 3 through 5 move the value in EMP-SS-NUM to the remaining partial key field, EMP-SS-NUM-0415.

Example 5

```
 PATH-- JOB
 010    WORK-FIELD  'DATA ENTRY CLERK    '

 017    CALL DB-EXIT ('FORMAT-6','JOB ','JOB-TITLE-NDX ',
*                     TITLE-0440,WORK-FIELD,20)
 017    RELS
 017    IDMS-STATUS EQ '0000' STOP-RUN
```

The DB-EXIT call generates an OBTAIN JOB WITHIN JOB-TITLE-NDX USING TITLE-0440 database retrieval command. In this example, ARG2 specifies the name of the indexed record, ARG3 specifies the name of the indexed set, and ARG4 identifies the name of the index key. ARG5 identifies an alphanumeric work field that contains the value of the indexed field; the 20-byte length of the work field (ARG5) is equal to the length of the index key.

Example 6

```
 PATHAA EMPLOYEE
 PATH-- DEPARTMENT

 017    CALL DB-EXIT ('OWNER' 'DEPARTMENT ','DEPT-EMPLOYEE ')
```

Each time CA Culprit executes the DB-EXIT call, it retrieves a DEPARTMENT record occurrence that is the owner record of the EMPLOYEE record occurrence currently in the input buffer.

Example 7

```
 PATHAA EMPLOYEE
 PATH-- JOB

 017    CALL DB-EXIT ('DBKEY' 'JOB ' X'01256902')
```

The DB-EXIT call generates an OBTAIN JOB DBKEY command. ARG1 specifies the type of call enclosed in single quotation marks; ARG2 specifies the record name JOB followed by a space and enclosed in single quotation marks. ARG3 specifies the db-key of the record to be retrieved as a hexadecimal literal. The first three bytes of the literal specify the page number, 75113, and the last two bytes specify the line number, 2.

Example 8

```
 PATH-- EMP-JOB-LR WHERE EMP-ID-0415 EQ EMP-ID-0415(1)
 GW0    INDEX
 GWO    EMP-ID.4 0301 0091 0054 0000

 017010 INDEX + 1 INDEX
 017    EMP-ID.INDEX EQ 0000 STOP-RUN
 017020 CALL DB-EXIT ('SET-VALUE' EMP-ID-0415 EMP-ID.INDEX
*                    4 '2')
 017    CALL DB-EXIT ('LR-NEXT' 'EMP-JOB-LR ')
 017    LR-STATUS EQ 'LR-NOT-FOUND' 010
 017    RELS
 017    B 020
```

This example includes two DB-EXIT calls. The first call instructs CA Culprit to set a value in the EMP-ID-0415 field in the input buffer. This field is qualified by a 1 in the WHERE clause on the dummy PATH parameter. The value set in the field is one of the values specified for global work field EMP-ID.

The second DB-EXIT call instructs CA Culprit to obtain the logical record occurrence that specifies an employee id value equal to the value in the input buffer. CA Culprit continues to retrieve the next logical record until it encounters LR-NOT-FOUND. The run terminates when the value of EMP-ID is 0000.

Appendix M: Profile Options

This section provides information about the keywords and operands for CA Culprit profile options. The CA Culprit profile options determine the report format, date format, and other site-specific characteristics.

For more information about the profile options or overriding the options at runtime, refer to the chapter PROFILE Parameter (see page 29).

CA Culprit uses default values for the profile options after installation. To change the installation default refer to the section Update Installation Default Profile Options.

This section contains the following topics:

Keywords and Operands

Block/Track Option

(z/VSE systems only) The BT option specifies the number of blocks per track for CA Culprit work files written to disk.

```
BT={ 1 ◄─
     2
     4
     8}
```

CALC Key Sign Option

The CALSIN option specifies whether to store IDMS CALC keys with an F or C sign.

```
CALSIN={F ◄─
       C }
```

- F-Specifies packed decimal IDMS CALC keys are stored with an F sign.
- C-Specifies packed decimal IDMS CALC keys contain a C sign.

Headers Option

The CCH option specifies whether to force headers.

```
CCH={ N ◄─
      A
      G }
```

- N-Specifies no headers are forced when a detail or total lines specifies a carriage control of 1. Headers print only when the specified lines per page for the report is exceeded or when a control break occurs for a sort field with a break code on 1.
- A-Specifies headers are always printed whenever a detail line or total line specify a carriage control of 1.
- G-Specifies headers are forced by a line printing with a carriage control of 1 only if it is a total line and the totals being printed are grand totals.

Assembly Date Option

The DATE option specifies the PROFILE assembly date.

DATE=*mm/dd/yy*

mm/dd/yy

Specifies the date the PROFILE CSECT was assembled.

This option should always be specified for diagnostic purposes.

Buffer Size Option

The DICTB option specifies report buffer size.

DICTB=*buffer-size-n*

buffer-size-n

Specifies the record size that is used for reports generated for the Data Dictionary Reporter and for IDMS-DC reports.

Default: 1000

Maximum: 32767

Lines Per Page Option

The DLLP option specifies the number of lines per page on each diagnostic list.

DLLP=*lines-per-page-n*

lines-per-page-n-

Specifies the number of lines per page that prints in the CA Culprit diagnostic listings. This option does not affect the number of lines per page in output reports.

Default: 55

Date Stamp Option

The DS option specifies the format of the date stamp that appears on CA Culprit listings and user output reports.

```
DS= { A
      E
      C
      H )
```

Note: For more information about date stamp see the chapter PROFILE Parameter (see page 29).

Error Options

The following error options specify the number of errors to be reported and the manner in which errors are processed. If this option is used in the MACRO, the option must be entered as indicated in the syntax shown below:

```
{ EE=
  IE=
  ME=      (error-count-n, return-code-n {I  {D )
  OE=                                    A   N}
  PE=                                    S}
  SE=  }
```

Note: For more information about error options see the chapter PROFILE Parameter (see page 29).

Report Error Level Option

The EX option specifies the error codes under which CA Culprit will read and report on the user's input files.

```
EX= { W
      E
      N }
```

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Hexadecimal Dump Option

The HD option specifies the format in which hexadecimal dumps issued by the extended error handling facility are to appear.

```
HD= { V
      H
      D }
```

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

IDMS Buffer Size Option

The IDMSB option specifies the buffer size for CA IDMS access.

IDMSB=*record-size-n*

record-size-n

Specifies the buffer size reserved for IDMS database accesses when the record size field is omitted from the INPUT parameter.

Default: 1000

Maximum: 32767

DDNAME Modification Option

(z/OS systems only) The IN0 and IN4 options are used to change CA Culprit step ddnames.

```
{ IN0=    (ddname,00,' ',00)
  IN4= }
```

ddname

Specifies the ddname to which the step is to be changed. The default is SYSIN.

- IN0 and IN4 specify the CA Culprit step whose ddname is changed as follows:
- IN0-CULP0 step
- IN4-CULP4 step

Report Lines Per Page Option

The LP option specifies the maximum number of lines per page for all user reports.

LP=lines-per-page-n

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Line Size Option

The LS option specifies the line size for all printed reports.

LS=*line-count-n*

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Numeric Editing Option

The MONEYED option specifies American or European numeric editing.

```
MONEYED= { A
           E }
```

- A-Specifies the default numeric editing is to be done in the American fashion, that is, decimal digits separated from integer digits with a period, and integer digits separated into groups of three with commas, for example, 1,000.52.
- E-Specifies European editing, that is, decimal digits separated from integer digits with a comma, and integer digits separated into groups of three with a period, for example, 1.000,52.

Tape Records/Block Option

(z/VSE systems only) The RB option specifies the size of CA Culprit work files written to tape.

```
RB= { 8 ◄─
      4
      2
      1 }
```

RB= (records/blocks)

Specifies the size of CA Culprit work files written to tape, expressed in 780-byte records per block.

Return Codes Option

The RC option specifies the return codes associated with the four levels of CA Culprit messages.

RC=(*return-code-n*..)

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Relocating Loader Option

(z/VSE systems only) The RELO option specifies if a relocating loader is present.

```
RELO= { Y ◄─
        N }
```

Y

Specifies the operating system has a relocating loader.

N

Specifies the operating system does not have a relocating loader.

Schema Name Option

The SCHMREQ option specifies whether a schema name is required.

```
SCHMREQ= { Y
           N ◄─}
```

Y

Specifies the schema name is required. The schema version remains optional.

N

Specifies the user can omit coding the schema name when entering the SS= operand on the INPUT parameter for access of a CA IDMS database.

Repeat First Page Option

The SF option specifies whether to repeat the first page of printed output on special forms.

```
SF= { N
      E }
```

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

SPIE/STXIT Routine Option

The SPIE option specifies whether to enable the SPIE/STXIT routine.

SPIE= { Y ◀–

N }

Y

Specifies CA Culprit extended error handling SPIE/STXIT routine is enabled.

N

Specifies the SPIE/STXIT routine is disabled. Program check errors that normally are trapped by CA Culprit result in immediate abnormal termination of the job.

File Characteristics Option

The S work-file-n option specifies the file whose default characteristics are to be overridden.

S work-file-n=(*filename, logical-unit-n, file-type, dtf-code*) -

work-file-n

Specifies the SYS number of a CA Culprit work, input, or key file whose default characteristics are to be overridden.

The following are the valid codes:

- 2
- 3
- 4
- 5
- 6

- 7
- 8
- 10

These codes correspond to the name SYS002, SYS003, and so on.

Filename

Specifies the filename (z/VSE) orddname (z/OS) to be used for this file.

logical-unit-n

Specifies the logical unit number associated with this file. Valid values are from 1 through 256.

z/OS users must code a value to satisfy the MACRO, but CA Culprit ignores the value.

file-type

Specifies the type of file. Valid codes are T (tape), D (disk), and blank enclosed in single quotation marks (indicates a LUB/PUB lookup).

z/OS users must code a value to satisfy the MACRO, but CA Culprit ignores the value.

dtf-code

A two-character hexadecimal code representing the one-byte DTF code of the device for the file. FF (the default) indicates that a LUB/PUB lookup is performed.

z/OS users must code a value to satisfy the MACRO, but CA Culprit ignores the value.

Time Stamp Option

The TS option specifies whether a time stamp is to appear on parameter listings and user reports. If this option is specified, all three values must be included, as indicated by the following syntax:

```
TS=( { Y   }, { Y   }, { Y   } )
       N ←       N ←       N ←
```

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Separator Character Option

The TSEP option specifies the hour/minute separator.

```
TSEP= { : ←
        / }
```

:

Specifies to use a colon to separate hours from minutes on a time stamp.

/

Specifies to use a slash to separate hours from minutes on a time stamp.

Source Library Option

The PARMLIB option specifies the source library that CA Culprit uses for =COPY and =MACRO parameters.

PARMLIB=*source-library-a*

For **source-library-a**, specify one of the following values:

STANDARD

Specifies the source library is a source statement library (z/VSE) or a partitioned data set (z/OS).

PANVALET

Specifies the source library is a CA Panvalet for z/OS library or a CA Panvalet for z/VSE library.

LIBRARIAN2

Specifies the source library is a CA Librarian for z/OS r2 library or a CA Librarian for z/VSE r2 library.

LIBR30

Specifies the source library is a CA Librarian for z/OS r3 library or a CA Librarian for z/VSE r3 library.

LIBRARIAN3

Specifies the source library is a CA Librarian for z/OS r3.1 and above library or a CA Librarian for z/VSE r3.1 and above library.

CA Panvalet File Option

(z/VSE systems only) The PANFILE option identifies the logical unit number and devices that CA Culprit uses for accessing a CA Panvalet for z/VSE library with =COPY and =MACRO parameters.

```
PANFILE='logical-unit-number=device-type1-a,device-type2-a'
```

Default: PANFILE='SYS006=333-,3330'

Note: The single quotation marks are required.

Social Security Number Format Option

The SSN option specifies the format of the social security number when the FS edit mask is used.

```
SSN= { D □
       A
       C }
```

Note: For more information about this option see the chapter PROFILE Parameter (see page 29).

Records Written Message Option

The RECMSG option specifies whether the message C750009 "Records Written for Report" is issued or not in the report output.

```
RECMSG= { Y □
          N }
```

Y

Specifies that the message is issued.

N

Specifies that the message is suppressed.

CULLUS12 Default Year Option

TheUS12YR option specifices the year from which the century is determined in user module CULLUS12 .

US12YR= *nn*

nn

When using an output format code which requires a century (ccyy) and the input date does not include one century will default to:

- 20 if yy <= nn
- 19 if yy > nn

Default: 40

Output DDNAME Override Option

Important: This option is for z/OS systems only.

When using one-step JCL, theOUTDD option changes the ddname used for the report output from the output phase (CULE). This option would be used to separate the report output from the listings and diagnostic messages of the CA Culprit run.

OUTDD= (*ddname*,00,' ',00)

ddname

Specifies the ddname to which the CULE step output is to be changed. If not specified, the dddname defaults to blanks and the SYS004 specification will be used.

If specified, a new DD statement must be added to the jcl to describe the ddname. The new DD statement must contain DCB=BLKSIZZE information or the job will abend with an S013. All reports in the same CA Culprit run will be written out to the new ddname unless a special file-type indicator (PS, IS, or NS) is specified on the output card

Update Installation Default Profile Options

At some point after the completion of the CA Culprit installation, you might have to modify the default Profile options.

You can create a profile module using the CULMPROF macro. Link the result into your custom load library as a stand-alone module named CULPPROF. Culprit dynamically loads this module at runtime.

z/OS Operating System Procedures

To create a Culprit profile module in z/OS, execute the z/OS Assemble and Link-edit JCL. Before you execute the JCL, substitute the name of your CULPPROF source member and insert the following binder statements:

```
SETOPT PARM(AMODE=24,RMODE=24)
NAME CULPPROF(R)
```

z/VSE Operating System Procedures

To create a Culprit profile module in z/VSE, execute the z/VSE Assemble and Link-edit JCL. Before you execute the JCL, substitute the name of your CULPPROF source member and insert the following link-edit statements:

```
PHASE   CULPPROF,*
INCLUDE CULPPROF
```

Index

=

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z