Advantage™ VISION:Inform® for CICS®

Utilities Guide

4.0

Computer Associates®

IFUTM040.PDF/D39-020-011

Contents

Chapter 12: Library Restore Utility (LBRESTOR)

Chapter 13: Promote Process Utility

Chapter 14: Query Migration Utility (LIBCOPY)

Chapter 15: VISION:Builder Quick Start Utility (BUILDRQS)

Chapter 1

Utilities Overview

The utilities delivered with the Advantage™ VISION:Inform® system are provided for the convenience of the system administrator. The VISION:Inform system does not depend on the utilities.

The VISION:Inform installation process installs the sample JCL for the utilities in the INFORM.JCL library.

This chapter contains a brief description of each of the utilities included with VISION:Inform. The utilities are listed in alphabetical order.

COBOL Quick Start Utility

This utility generates skeletal VISION:Inform file definitions from existing COBOL copybooks. You can process copybooks residing in standard MVS® data sets, CA-Panvalet libraries, or CA-Librarian libraries. The generated VISION:Inform definitions are saved in the VISION:Inform definition library. You edit generated file definitions with the Definition Processor.

Communication File Backup Utility

This utility creates a backup copy of the communication file on tape or disk. This backup of the communication file is a sequential data set containing the database queues, user queues, print queues, and log entries.

Communication File Purge Utility

This utility scans the communication file for queries and tasks that are in a READY status and purges those queries and tasks that are as old or older than a user specified date parameter (MM/DD/YYYY).

Communication File Restore Utility

This utility restores and reorganizes the communication file up to the last backup date. The utility deletes the old file, defines and initializes a new communication file, then reloads from the backup tape or disk. Items that you added to the old file after the last backup will be lost and need to be recreated.

DB2 Quick Start Utility

This utility generates VISION:Inform file definitions from existing DB2® table definitions. Each DB2 table that is processed becomes a segment in the VISION:Inform file definition that is being created.

Definition Convert Utility

This utility converts existing file definitions, table definitions, logical data view definitions, external requests, and ASL procedures into the format and order required for use with the Definition Processor. This utility is intended for customers who are upgrading from *Answer*/DB™ to VISION:Inform.

Field Description Merge Utility

This utility merges existing *Answer*/DB field descriptions, also known as FDHELPs, into existing VISION:Inform file definition source members. You can create and maintain field descriptions as part of the file definition. This utility is intended for customers who are upgrading from *Answer*/DB to VISION:Inform.

Glossary Utility

This utility produces standard glossary listings for table definitions, file definitions, and logical data view definitions.

Initialize Utility

This utility defines and initializes the foreground library, the background library, and the communication file. The utility uses the IDCAMS utility to define the VSAM cluster and allocate file space.

Library Backup Utility

This utility creates a backup copy of the background and foreground libraries. The resulting library backup tape contains file definitions, table definitions, alternate names, data views, procedures, and user profiles. The utility also saves QUERY and STMTS items under the user profiles.

Library Restore Utility

This utility restores and reorganizes the foreground library and the background library up to the last backup date. The utility deletes the old library, defines and initializes the new library, then reloads from the library backup tape.

Promote Process Utility

This utility maintains the foreground and background libraries. The utility is a REXX EXEC command procedure that migrates or promotes VISION:Inform definition items (file definitions, table definitions, and logical data view definitions) to the foreground and background libraries. Once you promote definitions, VISION:Inform uses them to process data extraction requests.

Query Migration Utility

This utility extracts items of type QUERY and STMTS from one VISION:Inform system (foreground library) and exports them to a second VISION:Inform system. This utility is for users needing to move queries from a test system to a production system without rekeying the queries into the target system.

VISION:Builder Quick Start Utility

This source statement retrieval utility retrieves definitions from the VISION:Builder® COMLIB common library or the VISION:Inform background library and populates a VISION:Inform definition library. The file definitions are already in the format used by VISION:Inform.

VISION:Inquiry Quick Start Utility

This utility generates VISION:Inform file definitions from existing VISION:Inquiry® file MAPGENs. The utility produces a separate VISION:Inform file definition from each VISION:Inquiry map and converts the VISION:Inquiry map information into file, segment, and field information in the file definition.

VISION:Results Quick Start Utility

This utility generates VISION:Inform file definitions from existing VISION:Results™ and VISION:Eighty™ file definitions.

Contacting Computer Associates

For further technical assistance with this product, please contact Computer Associates Technical Support on the Internet at esupport.ca.com. Technical Support is available 24 hours a day, seven days a week.

Chapter 2 COBOL Quick Start Utility (COBOLQS)

The COBOL Quick Start Utility, COBOLQS, generates a skeletal VISION:Inform file definition from an existing COBOL file definition. Once a skeletal VISION:Inform file definition has been created, you can specify any additional file information with the Definition Processor.

The COBOL Quick Start Utility retrieves COBOL copybooks from MVS data sets, CA-Panvalet libraries, and CA-Librarian libraries. The COBOL Quick Start Utility supports most VISION:Inform file definition types.

Utility Component

The COBOL Quick Start Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(COBOLQS)

This sample JCL to execute the COBOL Quick Start Utility is in the VISION:Inform JCL library, member COBOLQS.

Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

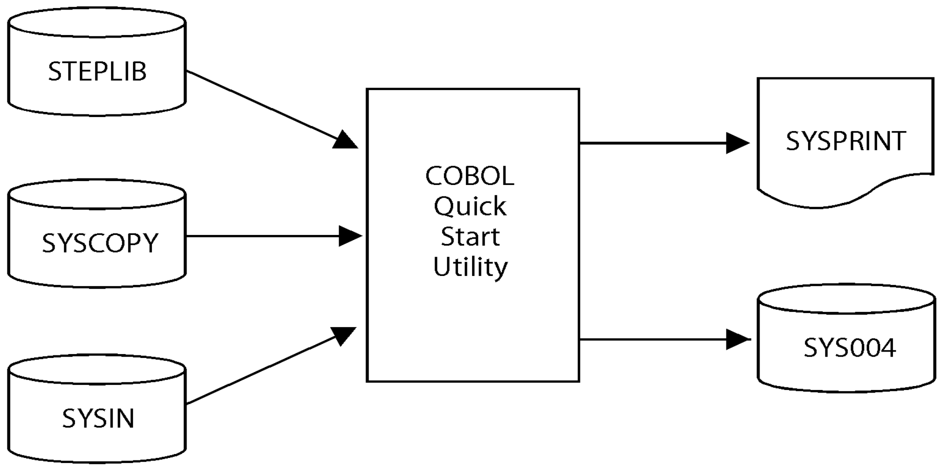

Figure 2-1 COBOL Quick Start Utility Flow Diagram

The COBOL Quick Start Utility uses the following input data sets:

STEPLIB	Required. Specifies the VISION:Inform installation load library.
SYSCOPY	Required. Specifies an MVS COBOL copybook library.
SYSIN	Required. Provides the appropriate COBOL Quick Start Utility control statements. You can also specify instream COBOL field definitions here. Use valid COBOL statements, containing no statements or characters that would cause COBOL compiler errors.

The COBOL Quick Start Utility produces two output files:

SYSPRINT	Contains a report on the file definition generation process which includes a listing of the COBOL statements that were processed.
SYS004	Contains the generated VISION:Inform file definition source statements.

Note: If you use a CA-Panvalet copybook library, specify a PANDD1 DD statement.

If you use a CA-Librarian copybook library, specify a MASTER DD statement.

See the *Advantage VISION:Inform Installation Guide for CICS* for more information.

Executing the Utility

COBOL Quick Start Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(COBOLQS)

The JCL contains an instream procedure, followed by the steps to execute the procedure. Notice that this JCL uses sample SYSIN data. Use this sample COBOL data to see how the COBOL Quick Start Utility works.

Note: You can also run this utility interactively under TSO/ISPF using the Definition Processor Main Menu Import option.

Specifying the Execution JCL

Figure 2-2 shows the sample JCL contained in INFORM.JCL in the member COBOLQS.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply values for the following procedure variables—for information, see Specifying Procedure Variables.
3. Provide a DD statement override for COBOLQS.SYSIN. This data set contains the required input control statements to generate the VISION:Inform file definition. For detailed information on these control statements, see Specifying Control Statements.
4. Make additional modifications, as required, to conform to your installation's standards.

The following figure shows the JCL to run the COBOL Quick Start Utility.

```
//* MEMBER COBOLQS                                                      00010000
//******************************************************************** 00020000
//* EXECUTE THE COBOL QUICK START UTILITY.                            * 00030000
//* ***** NOTE *****                                                  * 00040000
//* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES.      * 00050000
//* THE PANDD1  DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES. * 00060003
//* THE MASTER  DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES.* 00070003
//*                                                                   * 00080003
//* THIS UTILITY MAY ALSO BE INVOKED INTERACTIVELY UNDER TSO/ISPF     * 00090003
//* USING THE DEFINITION PROCESSOR COMPONENT IMPORT FUNCTION.         * 00100003
//******************************************************************** 00110000
//COBOLQS PROC LOADLIB=,                                                00120000
//           COPYLIB=,                                                  00130001
//           DEFLIB=,                                                   00140001
//           MEMBER=                                                    00150001
//COBOLQS EXEC PGM=COBOLQS,REGION=1024K                                 00160003
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00170002
//SYSPRINT DD SYSOUT=*                                                  00180001
//SYSCOPY  DD DISP=SHR,DSN=&COPYLIB                                     00190002
//*PANDD1  DD DISP=SHR,DSN=&COPYLIB                                     00200002
//*MASTER  DD DISP=SHR,DSN=&COPYLIB                                     00210002
//SYS004   DD DISP=OLD,DSN=&DEFLIB(&MEMBER)                             00220002
```

Figure 2-2   Sample Execution JCL for the COBOL Quick Start Utility, COBOLQS (Page 1 of 2)

```
//SYSIN    DD DUMMY                                                     00230001
//       PEND                                                           00240001
//*******************************************************************   00250000
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.         *   00260001
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                        *   00270001
//*                                                                 *   00280000
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.              *   00290001
//*    COPYLIB - THE COBOL COPY LIBRARY.                            *   00300000
//*    DEFLIB  - THE INFORM DEFINITION LIBRARY.                     *   00310001
//*    MEMBER  - MEMBER NAME FOR THE DEFINITION YOU ARE GENERATING. *   00320000
//*                                                                 *   00330000
//*  YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE        *   00340000
//*  COBOLQS.SYSIN DD OVERRIDE STMT.                                *   00350000
//*******************************************************************   00360000
//QS      EXEC COBOLQS,                                                 00370000
//             LOADLIB='INFORM.LOADLIB',                                00380001
//             COPYLIB='COBOL.COPYBOOK',                                00390000
//             DEFLIB='INFORM.DEFLIB',                                  00400001
//             MEMBER='NEWNAME'                                         00410000
//COBOLQS.SYSIN DD *                                                    00420000
  FILEGEN NAME=NEWNAME,TYPE=FIXED,BUFFSIZE=80                           00430003
  SEGMENT NAME=OFFICE,NUMBER=10,LEVEL=1                                 00440000
  $COBOL                                                                00450000
        01  OFFICE-DATA.                                                00460000
            02  OFFICE-CODE    PIC S9(3).                               00470000
            02  OFFICE-ADDRESS.                                         00480000
                03  OFFICE-STREET   PIC X(20).                          00490000
                03  OFFICE-CITY     PIC X(15).                          00500000
                03  OFFICE-STATE    PIC X(2).                           00510000
                03  OFFICE-ZIP.                                         00520000
                    04  OFFICE-ZIP-FIRST-FIVE  PIC X(5).                00530000
                    04  OFFICE-ZIP-LAST-FOUR   PIC X(4).                00540000
            02  OFFICE-PHONE       PIC 9(7)  OCCURS 3.                  00550000
            02  OFFICE-AREA-CODE   PIC X(3).                            00560000
            02  SPEED-DIAL         PIC X(3).                            00570000
            02  FILLER             PIC X(4).                            00580000
  $ECOBOL                                                               00590000
```

Figure 2-2 Sample Execution JCL for the COBOL Quick Start Utility, COBOLQS (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the variables for executing the COBOL Quick Start Utility JCL procedure, COBOLQS.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library (INFORM.LOADLIB).
COPYLIB	Specify the name of the COBOL copy library, (COBOL.COPYBOOK).
DEFLIB	Specify the VISION:Inform definition library (INFORM.DEFLIB) where the new file definition will be written. DEFLIB must specify a partitioned data set.
MEMBER	Specify a member name for the new file definition. Use the same name that is specified in the NAME parameter on the FILEGEN statement. ■ If you specify an existing member name, the utility replaces the existing member when the new definition is saved. ■ If you specify a new member name, the utility creates a new member.

Figure 2-3 JCL Procedure Variables for the COBOL Quick Start Utility

Using CA-Panvalet and CA-Librarian COBOL Copybooks

The COBOL Quick Start Utility provides direct access to COBOL copybooks that are stored in CA-Panvalet or CA-Librarian source libraries. Before using this utility, link edit the appropriate CA-Panvalet or CA-Librarian interface modules with the COBOL Quick Start Utility interface modules. The required interface modules are included in your CA-Panvalet or CA-Librarian software package. For details, see the *Advantage VISION:Inform Installation Guide for CICS*.

Specifying Control Statements

The COBOL Quick Start Utility uses the following control statements as input:

FILEGEN	Provides a name for the file definition that is being generated and identifies the type of file (that is, VSAM KSDS or IMS™) being generated.
SEGMENT	Defines the segments within a file.
$COBOL	Signals that an instream COBOL definition follows.
$ECOBOL	Signals the end of an instream COBOL definition.

The function and syntax of these statements are described in the following sections.

Coding Rules

Follow these rules when writing COBOL Quick Start Utility control statements:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement can contain keyword parameters. You can specify keyword parameters in any order. Separate the parameters by commas. Do not use embedded blanks between parameters.
- Specify each parameter unless stated otherwise.
- Place a comma following the last parameter on the statement to continue a statement on the next line.
- You can place comments after the last parameter with an intervening blank.

The utility scans columns 1 through 71 of the control statement and ignores columns 72 through 80.

FILEGEN Control Statement

FILEGEN NAME=*filedef*,TYPE=*filetype*, RECSIZE=*n*, RECBLK=*n*, BUFFSIZE=*n*, FLDPREFX=*pre*

FILEGEN — Required. FILEGEN is a control statement command that identifies the control statement type.

NAME — Required. The NAME parameter specifies the name of the file definition that is being generated.

- Begin the file name with an alphabetic letter.
- Specify the remaining characters as alphanumeric characters.
- Specify from 1 to 8 characters.

When the utility writes the generated file definition to a partitioned data set, the file name you specify here must be identical to the member name specified in the JCL (that is, the name following the MEMBER= parameter).

TYPE — Required. The TYPE parameter specifies the type of VISION:Inform file definition that you want to create. The following are valid file types.

File Type	Description
DB2	DB2 Relational Database
KSDS	VSAM Key Sequenced Data Set
ESDS	VSAM Entry Sequenced Data Set
AIX	VSAM Alternate Index Data Set
DLI	IMS Database
DLIHDAM	IMS HDAM Database
ISAMFIX	ISAM Fixed Length Record Format Data Set
ISAMVAR	ISAM Variable Length Record Format Data Set
FIXED	Fixed Length Record Format Data Set
VARIABLE	Variable Length Record Format Data Set
UNDEFINED	Undefined Record Format Data Set
GDBI	Generalized Data Base Interface Mapped File

RECSIZE — Optional. The RECSIZE parameter specifies the number of data bytes in the data portion of a record or segment. Enter a number from 1 to 9999. This parameter only applies to FIXED, ISAMFIX, VARIABLE, or ISAMVAR file types.

RECBLK	Optional. The RECBLK parameter specifies the number of records in each block. Enter a number from 1 to 9999. This parameter only applies to FIXED and ISAMFIX file types.
BUFFSIZE	Optional. The BUFFSIZE parameter specifies the size of the buffer needed to process the file. Enter a value from 1 to 32760 or 1K to 9999K. For DB2, DLI, DLIHDAM, and GDBI file types, enter the maximum amount of main storage required to hold a logical record. ■ For KSDS and ESDS file types, enter the maximum record size according to the VSAM cluster definition. Enter a value from 1K to 9999K for values greater than 32,767. ■ If the file type is AIX, enter the alternate index control interval size. For ISAMFIX, ISAMVAR, FIXED, VARIABLE, and UNDEFINED, enter the block size.
FLDPREFX	Optional. The FLDPREFX parameter specifies from 1 to 3 characters as a prefix for generating primary field names in the VISION:Inform file definition. Primary field names are required in a VISION:Inform file definition, must be unique, and can be from 1 to 8 characters. ■ Since COBOL field names can be longer than eight characters and are not necessarily unique within the first eight characters, the COBOL Quick Start Utility automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number. ■ If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999.

SEGMENT Control Statement

SEGMENT NAME=*name*, NUMBER=*n*, LEVEL=*n*, {COPYCOBOL=*copybookname*
| COPYPCOBOL=*copybookname* | COPYLCOBOL=*copybookname*}

SEGMENT	Required. SEGMENT is a control statement command that identifies the control statement type.
NAME	Required. The NAME parameter assigns a name to a segment. ■ Begin the segment name with an alphabetic letter. ■ Specify the remaining characters as alphanumeric. ■ Specify from 1 to 8 characters.
NUMBER	Required. The NUMBER parameter assigns a number that uniquely identifies the segment. ■ Enter a number from 1 to 255. ■ Make subordinate segments a number larger than the parent segment and smaller than any subordinate segments.
LEVEL	Required. The LEVEL parameter specifies the subordination of segments. ■ Set the root segment to have a level number of 1. ■ Assign all subordinate segments a number from 2 to 9.
COPYCOBOL	Optional. The COPYCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. Locate this copybook in an MVS data set assigned to the SYSCOPY DD statement in the JCL.
COPYPCOBOL	Optional. The COPYPCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. Locate this copybook in a CA-Panvalet library assigned to the PANDD1 DD statement in the JCL.
COPYLCOBOL	Optional. The COPYLCOBOL parameter specifies the name of the COBOL copybook that contains the field definitions for this segment. Locate this copybook in a CA-Librarian library assigned to the MASTER DD statement in the JCL.

Processing Notes

COPYCOBOL, COPYPCOBOL, and COPYLCOBOL parameters are mutually exclusive. Specify only one of these parameters in a SEGMENT statement. Figure 2-4 shows an example of the input control statements used with the COPY parameter.

```
FILEGEN NAME=XXXX,TYPE=X

SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X,

  COPYCOBOL=copybookname
```

Figure 2-4 SEGMENT Statement with the COPY Parameter

When you use any of the COPY parameters (COPYCOBOL, COPYPCOBOL, COPYLCOBOL), you can specify the NOPRINT option to suppress the listing of the copybook on the SYSPRINT file. Specify this option as:

```
COPYCOBOL=(copybookname,NOPRINT)
```

If you do not specify a COPY parameter on the SEGMENT statement, then provide an instream COBOL definition. In this case, follow the SEGMENT statement with a $COBOL statement, the instream COBOL definition, and the $ECOBOL statement, which marks the end of the definition. For more information see $COBOL and $ECOBOL Control Statements.

$COBOL and $ECOBOL Control Statements

Use the $COBOL and $ECOBOL control statements in place of the SEGMENT COPY parameter to process instream COBOL source statements.

- Place the $COBOL statement after a SEGMENT statement that does not contain a COPY parameter.
- Place the actual COBOL source statements to be processed after the $COBOL statement. Use the instream COBOL source statements to generate field definitions for the segment that preceded the $COBOL statement.
- Mark the end of the instream COBOL source statements by the $ECOBOL statement.

Place the $COBOL and $ECOBOL statements anywhere within columns 1 to 71 as the only command on the statement. There are no parameters for either statement.

```
FILEGEN NAME=XXXX,TYPE=X
SEGMENT NAME=XXXX,NUMBER=XX,LEVEL=X
$COBOL
instream COBOL source statements
$ECOBOL
```

Figure 2-5 $COBOL and $ECOBOL Control Statements

Conversion Rules

Editing the Generated VISION:Inform File Definition

The COBOL Quick Start Utility generates a skeletal VISION:Inform file definition which you edit and validate using the Definition Processor prior to use. Consider the following items when editing the generated file definition:

Segment Key Assignment	Each segment must define at least one field as the key. You can assign additional keys as needed.
Segment Information	Provide additional segment information such as segment order and number of fixed occurrences as needed.
Field Information	Modify primary and alternate field name assignments as needed (see the following). You can also provide additional field information such as rounding, editing, and automatic table lookup results as needed.

Specifying VISION:Inform Field Names

Assign a unique name to all fields within an VISION:Inform file definition. This name is referred to as the primary field name. Specify from 1 to 8 characters for the name.

When you build a skeletal VISION:Inform file definition from COBOL field definitions, the utility assigns a unique primary field name to each field. This field name is from 1 to 8 characters.

- Since COBOL field names can be longer than eight characters and are not necessarily unique within the first eight characters, the COBOL Quick Start Utility automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number.
- If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format Fnnnnnnn where nnnnnnn is a number from 0000001 to 9999999.

The COBOL Quick Start Utility uses the COBOL field name to generate the column heading specifications.

Once you create the skeletal file definition, you can replace the generated primary field names with more descriptive primary field names using the Definition Processor.

Unsupported COBOL Specifications

The COBOL Quick Start Utility does not support the following COBOL field specifications:

- 66 levels. All references to the field are removed. The utility issues a warning message.
- 77 levels. Any field defined at level 77 will have a starting location of 1. Check the resulting definition. The utility issues a warning message.
- 88 levels. All references to the field are removed. The utility issues a warning message.
- Numeric fields with more than 9 digits to the right of the decimal place are supported for packed fields; however, for any other field type, a warning message is issued indicating that the number of decimal places has been truncated to 9.
- Binary fields larger than S9(9) are generated as character fields. The utility issues a warning message.
- COMP-2 data types are generated as 8-byte character fields. The utility issues a warning message.
- If the length of a numeric field exceeds 15 digits, the field is generated as character. The utility issues a warning message.
- The P edit parameter on the PICTURE clause always generates a zero SCALE value. Check the resulting definition. The utility issues a warning message.
- If the size of a field exceeds 255, the utility issues a warning message indicating that the generated field length has been truncated to 255. Modify the generated definition to include an additional field that defines the remaining bytes or at least the last byte of the field.
- OCCURS clause. The utility issues a warning message to indicate that only one occurrence of the item was generated. Modify the definition to account for the additional occurrences.

  OCCURS Clause — Variably Occurring

  If the item on the OCCURS clause is variably occurring (DEPENDING ON clause present), then the additional occurrences can be defined by defining the generated occurrence as a lower level variably occurring segment, if possible. This requires the field containing the number of occurrences to be defined as a count field in the generated file definition.

  When the utility encounters a DEPENDING ON clause, it issues an additional message to inform you to adjust the field start locations of subsequent fields. This is because the generated start location of the next field will be incorrect if you define the generated occurrence as a lower level variably occurring segment.

Also, the start location of the next field is calculated using the maximum number of occurrences when in reality the number of occurrences varies.

OCCURS Clause — Fixed Occurring

If the item on the OCCURS clause is fixed occurring (no DEPENDING ON clause), then the additional occurrences can be defined either by:

- Defining the generated occurrence as a lower level fixed occurring segment. If this method is used, a reference to the field will initiate a loop where each occurrence of the field is automatically processed.
- Defining a separate field for each occurrence. If this method is selected, then a separate field name exists for each occurrence.
- Defining one field whose length accommodates all occurrences (that is, define a 30-byte field if you have a 3-byte field that occurs ten times). If this method is chosen, then each occurrence can be accessed by using dynamic partial fielding, which is similar to indexing.
- If the field name is FILLER or blank, the utility removes all references to the field from the definition.

Chapter

3

Communication File Backup Utility (CMBACKUP)

The Communication File Backup Utility, CMBACKUP, creates a backup copy of the communication file on tape or disk. This backup of the communication file is a sequential data set containing the database queues, user queues, print queues, and log entries.

Utility Component

The Communication File Backup Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(CMBACKUP)

This sample JCL to execute the Communication File Backup Utility is in the VISION:Inform JCL library, member CMBACKUP.

Flow Diagram

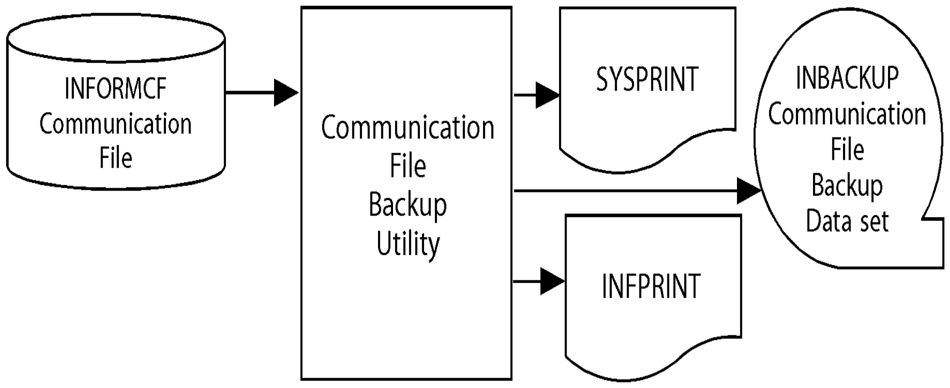
INFORMCF Communication File
Communication File Backup Utility
SYSPRINT
INBACKUP Communication File Backup Data set
INFPRINT

Figure 3-1 Communication File Backup Utility Flow Diagram

Utility Flow

The Communication File Backup Utility, CMBACKUP, consists of the following steps:

1. VER step

   The verify step executes the VSAM IDCAMS utility to verify the availability of the VSAM communication file to the backup job.

   The utility creates the SYSOUT data set, VERIFY.SYSPRINT.

   The acceptable return code for this step is 0.
2. BACKUP step

   This step creates a copy of the communication file.

   The utility creates the SYSOUT data set, BACKUP.INFPRINT.

   The acceptable return code for this step is 0.

Note: The step names correspond to JCL job step names.

Executing the Utility

The Communication File Backup Utility is a batch utility you run by submitting the JCL provided in:

```
INFORM.JCL(CMBACKUP)
```

The JCL contains an instream procedure, followed by the steps to execute this procedure.

Specifying the Execution JCL

Figure 3-2 shows the sample JCL contained in INFORM.JCL in the member CMBACKUP.

Review the JCL carefully. At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values. (See Specifying Procedure Variables.)
3. Make additional modifications, as necessary, to conform to your installation's standards.

The following figure shows the sample JCL to run the Communication File Backup Utility.

```
//* MEMBER CMBACKUP                                                     00010000
//********************************************************************* 00020000
//*   THIS PROCEDURE CREATES A BACKUP COPY OF THE COMMUNICATIONS      * 00030001
//*   FILE ON TAPE OR DISK, DEFAULTING TO TAPE.                       * 00040001
//********************************************************************* 00050000
//CMBACKUP PROC INFCOM=,                                                00060000
//             LOADLIB=,                                                00070000
//             NAME=,                                                   00080000
//             FILENUM=,                                                00090000
//             UNIT=,                                                   00100000
//             VOLSER=                                                  00110000
//VER     EXEC PGM=IDCAMS,REGION=512K                                   00120000
//INFCOM    DD DISP=SHR,DSN=&INFCOM                                     00130000
//SYSPRINT  DD SYSOUT=*                                                 00140000
//BACKUP  EXEC PGM=INFORMUU,REGION=512K                                 00150000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00160000
//INFPRINT  DD SYSOUT=*                                                 00170000
//INFORMCF  DD DISP=SHR,DSN=&INFCOM                                     00180000
//INBACKUP  DD DISP=(NEW,KEEP),DSN=&NAME,UNIT=&UNIT,VOL=SER=&VOLSER,    00190000
//             LABEL=(&FILENUM,SL)                                      00200000
//        PEND                                                          00210000
//********************************************************************  00220000
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THE PROCEDURE.            *  00230001
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          *  00240001
//*                                                                  *  00250000
//* INFCOM  - THE INFORM COMMUNICATIONS FILE.                        *  00260001
//* LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                  *  00270001
//* NAME    - THE BACKUP FILE TO CREATE.                             *  00280000
//* FILENUM - THE NUMBER OF THE TAPE FILE TO CONTAIN THE BACKUP.     *  00290000
//*           THE DEFAULT IS 1.                                      *  00300000
//* UNIT    - THE UNIT OF THE BACKUP FILE. THE DEFAULT IS TAPE.      *  00310000
//* VOLSER  - THE VOLUME SERIAL NUMBER OF THE BACKUP FILE.           *  00320000
//********************************************************************  00330000
//BACKUP  EXEC CMBACKUP,                                                00340000
//             INFCOM='INFORM.INFCOM',                                  00350001
//             LOADLIB='INFORM.LOADLIB',                                00360001
//             NAME='INFORM.BACKUP.INFCOM',                             00370001
//             FILENUM=1,                                               00380000
//             UNIT=TAPE,                                               00390000
//             VOLSER=XXXXXX                                            00400000
//VER.SYSIN DD *                                                        00410000
         VERIFY FILE(INFCOM)                                            00420000
```

Figure 3-2 Sample Execution JCL for the Communication File Backup Utility, CMBACKUP

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the Communication File Backup Utility JCL procedure, CMBACKUP.

Variable name	Description
INFCOM	Specify the name of the communication file to be backed up.
LOADLIB	Specify the name of the VISION:Inform installation load library to be used by the Communication File Backup Utility.
NAME	Specify the name of the communication file backup data set.
FILENUM	Specify the number of the tape file containing the communication file backup data. The default is 1.
UNIT	Specify the unit of the communication file backup data set. The default is tape.
VOLSER	Specify the volume serial number of the communication file backup data set.

Figure 3-3 JCL Procedure Variables for the CMBACKUP Utility

Generated Output

The Communication File Backup Utility outputs:

- The communication file backup data set.
- The output file, BACKUP.INFPRINT, summarizing the results of process. It contains a list of the members that were copied to the communication file backup data set.

Invalid Internal Pointers

The following message in the INFPRINT listing means that a member (queries, reports, and so on) has an invalid internal pointer.

```
JX01** Member ******** truncated due to bad pointer **.
```

If the Communication File Backup Utility issues this message, run the Library Restore Utility to fix the bad pointer. Note that the Communication File Restore Utility fixes broken pointers, but some data could be lost.

Space Utilization Statistics

Each time you run the VISION:Inform Communication File Backup Utility against the communication file, the utility generates space utilization statistics and writes data to the INFPRINT summary listing.

A sample of the actual message text follows. The asterisks in the message text indicate where the utility inserts numerical values for each run.

```
**S801 Of ******* overflow blocks, ******* are free, and ******* are in use.
**S802 Of ******* root blocks, ******* are free, and ******* are in use.
**S803 Of ******* root blocks, ******* have overflowed.
```

With these space utilization statistics, the system administrator can now determine:

- When to allocate additional space for the library (usually when the number of available overflow blocks becomes too small).
- When to increase the number of root blocks (usually when the number of available root blocks becomes too small).
- When to increase the block sizes (usually when the number of overflow blocks becomes too high).
- When to increase the bucket sizes (usually when the number of overflow blocks becomes too high).

Chapter 4

Communication File Purge Utility (PURGUTIL)

The Communication File Purge Utility, PURGUTIL, scans the communication file for queries and tasks that are in a READY status and purges those queries and tasks that are as old as, or older than, a specified date parameter in MM/DD/YYYY format.

Utility Components

The Communication File Purge Utility is composed of the following components:

Sample Execution JCL — INFORM.JCL(PURGUTIL)

This sample JCL to execute the Communications File Purge Utility is in the VISION:Inform JCL library, member PURGUTIL.

Sample Input Source — INFORM.SRCLIB

The installation process places following two members in the VISION:Inform source library:

- PURGLIST — This is the input file (INFIN) to STEP1 of the Communication File Purge Utility procedure.
- PURGPROG — This is the supplied source utility program. M4INPUT is used in STEP2 of the Communication File Purge Utility procedure.

Flow Diagram

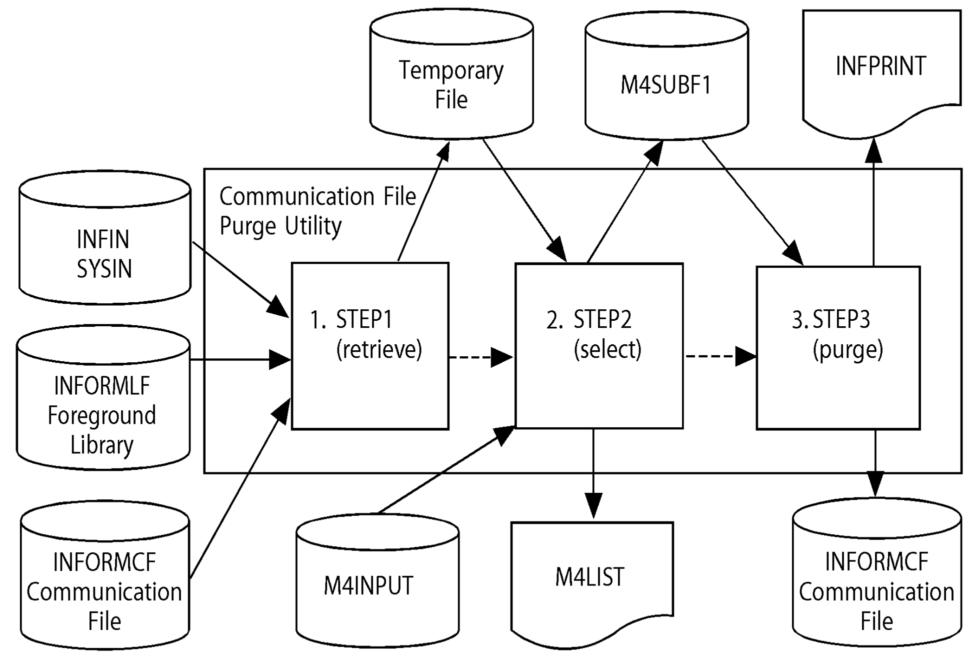
Temporary File
M4SUBF1
INFPRINT
Communication File Purge Utility
INFIN SYSIN
1. STEP1 (retrieve)
2. STEP2 (select)
3. STEP3 (purge)
INFORMLF Foreground Library
INFORMCF Communication File
M4INPUT
M4LIST
INFORMCF Communication File

Figure 4-1 Communication File Purge Utility Flow Diagram

Utility Flow

The Communication File Purge Utility consists of the following steps:

1. STEP1

   The retrieve step creates a temporary file (INFPRINT) that contains a list of queries and tasks that are in the ready status. INFPRINT is input to STEP2.

   The acceptable return code for this step is 0.

2. STEP2

   The select step checks the items passed by the retrieve step in the temporary file against the specified completion date parameter and selects the items that meet the purge criteria.

   The step creates the SYSOUT data set, STEP2.M4LIST.

   The acceptable completion code for this step is 0.

3. STEP3

   The purge step uses the temporary file created by the select step to purge the selected queries and tasks.

   The step creates the SYSOUT data set, STEP3.INFPRINT.

   The acceptable completion code for this step is 0.

Note: Utility flow step names correspond to JCL job step names.

Executing the Utility

The Communication File Purge Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(PURGUTIL)

This JCL contains an instream procedure, followed by the steps to execute this procedure.

Specifying the Execution JCL

Figure 4-2 shows the sample JCL contained in INFORM.JCL in the member PURGUTIL.

Note: Depending on the share options set in the INIT job, you may have to take the foreground library offline or bring down CICS® to run this utility.

Review the JCL carefully. At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

Note: If the IBM® Language Environment® load library is not in the standard load library list made available to all batch programs on your host system, concatenate it to the STEP2.STEPLIB DD statement.

The following figure shows the JCL to run the Communication File Purge Utility.

```
//* MEMBER PURGUTIL                                                      00010000
//********************************************************************* 00020000
//* THIS PROCEDURE RUNS THE COMMUNICATIONS FILE PURGE UTILITY. A DATE * 00030000
//* PARAMETER IS PASSED TO THE UTILITY. ALL 'READY' STATUS REPORTS    * 00040000
//* WITH A COMPLETION DATE EQUAL TO OR OLDER THAN THE SPECIFIED DATE  * 00050000
//* WILL BE PURGED FROM THE COMMUNICATIONS FILE.                      * 00060000
//* *** NOTE ***  IF THE LANGUAGE ENVIRONMENT LOAD LIBRARY IS NOT IN  * 00070002
//*               THE STANDARD LOAD LIBRARY LIST MADE AVAILABLE TO    * 00080002
//*               ALL BATCH PROGRAMS ON YOUR HOST SYSTEM, IT MUST BE  * 00090002
//*               CONCATENATED TO THE STEP2.STEPLIB DD STATEMENT.     * 00100002
//* *** ALSO ***  DEPENDING ON THE SHARE OPTIONS SET IN THE INIT JOB, * 00110002
//*               CICS MAY HAVE TO BE DOWN TO RUN THIS UTILITY.       * 00110012
//********************************************************************* 00120000
//PRGUTIL PROC RGN=,                                                     00130000
//            LOADLIB=,                                                  00140000
//            SORTLIB=,                                                  00150000
//            SRCLIB=,                                                   00160000
//            DATE=,                                                     00170000
//            UTLIB=,                                                    00180000
//            FGLIB=,                                                    00190000
//            INFCOM=                                                    00200001
```

Figure 4-2 Sample Execution JCL for the Communication File Purge Utility, PURGUTIL (Page 1 of 2)

```
//STEP1   EXEC PGM=INFORMSB,REGION=&RGN                                 00210001
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00220000
//INFORMLF DD DISP=SHR,DSN=&FGLIB                                       00230001
//INFORMCF DD DISP=SHR,DSN=&INFCOM                                      00240001
//INFPRINT DD DISP=(NEW,PASS),DSN=&&PRG,UNIT=SYSDA,SPACE=(TRK,(10,10))  00250001
//INFLIST   DD DUMMY                                                    00260001
//INFREPT   DD DUMMY                                                    00270001
//INFIN     DD DISP=SHR,DSN=&SRCLIB(PURGLIST)                           00280001
//STEP2   EXEC PGM=MARKIV,REGION=&RGN,PARM='&DATE'                      00290000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00300000
//M4LIST    DD SYSOUT=*                                                 00310000
//M4REPO    DD DISP=(NEW,PASS),DCB=(BLKSIZE=6400,BUFNO=3),              00320000
//             UNIT=SYSDA,SPACE=(TRK,(5,2),RLSE)                        00330000
//M4SORT    DD DISP=(NEW,PASS),UNIT=SYSDA,SPACE=(TRK,1)                 00340000
//SYSOUT    DD DUMMY                                                    00350000
//SORTLIB   DD DISP=SHR,DSN=&SORTLIB                                    00360000
//SORTWK01 DD SPACE=(CYL,1,,CONTIG),UNIT=SYSDA                          00370000
//SORTWK02 DD SPACE=(CYL,1,,CONTIG),UNIT=SYSDA                          00380000
//SORTWK03 DD SPACE=(CYL,1,,CONTIG),UNIT=SYSDA                          00390000
//M4LIB     DD DISP=SHR,DSN=&UTLIB                                      00400000
//M4OLD     DD DSN=&&PRG,DISP=(OLD,PASS)                                00410000
//M4SUBF1   DD DSN=&&STMTS,DISP=(,PASS),UNIT=SYSDA,SPACE=(TRK,(10,5))   00420000
//M4INPUT   DD DISP=SHR,DSN=&SRCLIB(PURGPROG)                           00430000
//STEP3   EXEC PGM=INFORMSB,REGION=&RGN                                 00440001
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00450000
//INFORMLF DD DISP=SHR,DSN=&FGLIB                                       00460001
//INFORMCF DD DISP=SHR,DSN=&INFCOM                                      00470001
//INFPRINT DD SYSOUT=*                                                  00480001
//INFLIST   DD DUMMY                                                    00490001
//INFREPT   DD DUMMY                                                    00500001
//INFIN     DD DISP=(OLD,PASS),DSN=&&STMTS                              00510001
//        PEND                                                          00520000
//******************************************************************** 00530000
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           * 00540000
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          * 00550000
//*                                                                   * 00560000
//* RGN     - THE REGION SIZE (DEFAULT 1800K).                        * 00570000
//* LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                   * 00580001
//* SORTLIB - THE LOAD LIBRARY CONTAINING THE SYSTEM SORT PROGRAM.    * 00590001
//*           THE DEFAULT IS SYS1.SORTLIB.                            * 00600001
//* SRCLIB  - THE INFORM INSTALLATION SOURCE LIBRARY.                 * 00610001
//* DATE    - THE DATE PARAMETER FOR THE PURGE UTILITY. ALL 'READY'   * 00620000
//*           STATUS REPORTS WITH A COMPLETION DATE EQUAL TO OR       * 00630000
//*           OLDER THAN THIS DATE WILL BE PURGED.                    * 00640000
//*           SPECIFY THE DATE IN MM/DD/YYYY FORMAT.                  * 00650002
//* UTLIB   - THE INFORM INSTALLATION UTILITY LIBRARY.                * 00660001
//* FGLIB   - THE INFORM FOREGROUND LIBRARY.                          * 00670001
//* INFCOM  - THE INFORM COMMUNICATION FILE.                          * 00680001
//******************************************************************** 00690000
//INF     EXEC PRGUTIL,RGN=1800K,                                       00700001
//             LOADLIB='INFORM.LOADLIB',                                00710001
//             SORTLIB='SYS1.SORTLIB',                                  00720000
//             SRCLIB='INFORM.SRCLIB',                                  00730001
//             DATE='MM/DD/YYYY',                                       00740002
//             UTLIB='INFORM.UTLIB',                                    00750001
//             FGLIB='INFORM.FGLIB',                                    00760001
//             INFCOM='INFORM.INFCOM'                                   00770001
```

Figure 4-2 Sample Execution JCL for the Communication File Purge Utility, PURGUTIL (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the procedure variables you specify when executing the supplied Communication File Purge Utility JCL procedure, PURGUTIL.

Variable name	Description
FGLIB	Specify the name of the foreground library.
UTLIB	Specify the name of the VISION:Inform utility library.
LOADLIB	Specify the name of the VISION:Inform installation load library to be used by the utility.
SRCLIB	Specify the name of the VISION:Inform installation source library.
INFCOM	Specify the name of the VISION:Inform communication file.
DATE	The date parameter for the Communication File Purge Utility. All READY status queries or tasks with a completion date equal to or later than this date will be purged. The date is specified in MM/DD/YYYY format.
SORTLIB	Specify the name of the load library containing the system SORT program (default SYS1.SORTLIB).

Figure 4-3 JCL Procedure Variables for the Communication File Purge Utility

Generated Output

The Communication File Purge Utility produces the following informational listings:

STEP2.M4LIST	This contains a list of the items that have met the date purge criteria.
STEP3.INFPRINT	This listing summarizes the results and contains a list of the members that were purged from the communication file.

Chapter 5

Communication File Restore Utility (CMRESTOR)

The Communication File Restore Utility, CMRESTOR, restores and reorganizes the communication file up to the last backup date. The utility deletes the old file, defines and initializes a new communication file, then restores the contents from the latest communication file backup tape or disk. If you have added items after the last backup, you need to recreate them.

Use the sequential data set created by the Communication File Backup Utility as input to the Communication File Restore Utility.

Note: Take the communication file offline to run this utility.

Utility Component

The Communication File Restore Utility is made up of the following component:

Sample Execution JCL — INFORM.JCL(CMRESTOR)

This sample JCL to execute the Communication File Restore Utility is in the VISION:Inform JCL library, member CMRESTOR.

Flow Diagram

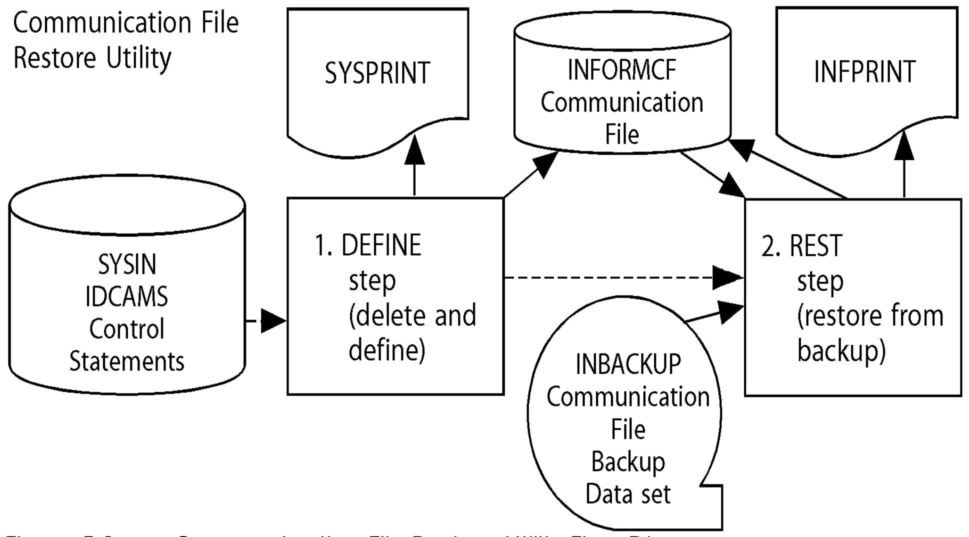
Communication File
Restore Utility
SYSPRINT
INFORMCF Communication File
INFPRINT
SYSIN IDCAMS Control Statements
1. DEFINE step (delete and define)
2. REST step (restore from backup)
INBACKUP Communication File Backup Data set

Figure 5-1 Communication File Restore Utility Flow Diagram

Utility Flow

The Communication File Restore Utility, CMRESTOR, consists of the following steps:

1. DEFINE step

   The define step executes the VSAM IDCAMS utility to delete and define the communication file.

   The utility creates the SYSOUT data set, DEFINE.SYSPRINT.

   The acceptable return code for this step is 0.

2. REST step

   The restore step initializes and restores the communication file.

   The utility creates the SYSOUT data set, REST.INFPRINT.

   The acceptable return code for this step is 0.

   A return code of 08 results if the restore step is executed for a new communication file, such as in the release upgrade process.

Note: The step names correspond to JCL job step names.

Executing the Utility

The Communication File Restore Utility, CMRESTOR, is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(CMRESTOR)

This JCL contains an instream procedure, followed by the steps to execute this procedure.

Specifying the Execution JCL

Figure 5-2 shows the sample JCL contained in INFORM.JCL in the member CMRESTOR. Review the JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL for the Communication File Restore Utility.

```
//* MEMBER CMRESTOR                                                       00010000
//********************************************************************** 00020000
//* THIS PROCEDURE RESTORES THE SYSTEM COMMUNICATIONS FILE FROM        * 00030000
//* THE BACKUP DATASET CREATED BY THE 'CMBACKUP' JOB. THIS JOB         * 00040000
//* ASSUMES THAT THE BACKUP FILE IS ON TAPE.                           * 00050000
//********************************************************************** 00060000
//CMRESTOR PROC INFCOM=,                                                  00070000
//             LOADLIB=,                                                  00080000
//             NAME=,                                                     00090000
//             FILENUM=,                                                  00100000
//             UNIT=,                                                     00110000
//             VOLSER=                                                    00120000
//DEFINE  EXEC PGM=IDCAMS,REGION=512K                                     00130000
//SYSPRINT  DD SYSOUT=*                                                   00140000
//REST    EXEC PGM=INFORMUL,REGION=512K                                   00150000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                      00160000
//INFPRINT  DD SYSOUT=*                                                   00170000
//INFORMCF  DD DISP=SHR,DSN=&INFCOM                                       00180000
//INBACKUP  DD DISP=OLD,DSN=&NAME,UNIT=&UNIT,VOL=SER=&VOLSER,             00190000
//             LABEL=(&FILENUM,SL)                                        00200000
//        PEND                                                            00210000
//*********************************************************************  00220000
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.            *  00230002
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                           *  00240001
//*                                                                   *  00250000
//* INFCOM  - THE INFORM COMMUNICATIONS FILE.                         *  00260001
//* LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                   *  00270001
//* NAME    - THE BACKUP FILE CREATED BY THE BACKUP PROCEDURE.        *  00280000
//* FILENUM - THE NUMBER OF THE TAPE FILE CONTAINING THE BACKUP.      *  00290000
//*           THE DEFAULT IS 1.                                       *  00300000
//* UNIT    - THE UNIT OF THE BACKUP FILE. THE DEFAULT IS TAPE.       *  00310000
//* VOLSER  - THE VOLUME SERIAL NUMBER OF THE BACKUP FILE.            *  00320000
//*                                                                   *  00330000
//*   ALSO CHANGE THE DEFINE.SYSIN SPECIFICATION AS YOU DID FOR THE   *  00340000
```

Figure 5-2 Sample Execution JCL for the Communication File Restore Utility, CMRESTOR (Page 1 of 2)

```
//*   INIT PROCEDURE.                                                  *  00350000
//*                   NAME('INFORM.INFCOM')                            *  00360001
//*                   NAME('INFORM.INFCOM.DATA')                       *  00370001
//*                   VOL('VOLUME')                                    *  00380000
//*                   OWNER('USER')                                    *  00390000
//*                                                                    *  00400000
//*   REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR MORE          *  00410000
//*   INFORMATION.                                                     *  00420000
//*********************************************************************  00430000
//RESTORE EXEC CMRESTOR,                                                  00440000
//             INFCOM='INFORM.INFCOM',                                    00450001
//             LOADLIB='INFORM.LOADLIB',                                  00460001
//             NAME='INFORM.BACKUP.INFCOM',                               00470001
//             FILENUM=1,                                                 00480000
//             UNIT=TAPE,                                                 00490000
//             VOLSER=XXXXXX                                              00500000
//DEFINE.SYSIN DD *                                                       00510000
      DELETE 'INFORM.INFCOM' CLUSTER PURGE                                00520001
      DEFINE CLUSTER -                                                    00530000
               (NAME('INFORM.INFCOM') -                                   00540001
               VOL('VOLUME') -                                            00550000
               RECORDS(409) -                                             00560000
               SHAREOPTIONS(3 3) -                                        00570000
               WRITECHECK NUMBERED -                                      00580000
               OWNER('USER')) -                                           00590000
             DATA(NAME('INFORM.INFCOM.DATA') -                            00600001
               CONTROLINTERVALSIZE(4096) -                                00610000
               RECORDSIZE(4088 4088) -                                    00620000
               BUFFERSPACE(8192))                                         00630000
```

Figure 5-2 Sample Execution JCL for the Communication File Restore Utility, CMRESTOR (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the Communication File Restore Utility JCL procedure, CMRESTOR.

Variable name	Description
INFCOM	Specify the name of the communication file to be restored.
LOADLIB	Specify the name of the VISION:Inform installation load library to be used by the utility.
NAME	Specify the name of the data set that contains the communication file backup data.
FILENUM	Specify the number of the tape file containing the communication file backup data set.
UNIT	Specify the unit of the communication file backup data set. The default is tape.
VOLSER	Specify the volume serial number of the communication file backup data set.

Providing DD Statement Overrides

Provide the following DD statement overrides when executing the Communication File Restore Utility.

In the following example, you modify the text in lowercase letters. Do not change the text in uppercase letters.

DEFINE.SYSIN for Communication File Delete and Define

The following figure shows sample input to the Communication File Restore Utility.

```
DELETE 'inform.infcom' CLUSTER PURGE
DEFINE CLUSTER -
               (NAME('inform.infcom') -
               VOL('volume') -
               RECORDS(nnn) -
               SHAREOPTIONS(2 3) -
               WRITECHECK NUMBERED -
               OWNER('user')) -
               DATA(NAME('inform.infcom.DATA') -
               CONTROLINTERVALSIZE(4096) -
               RECORDSIZE(mmm mmm))
```

Figure 5-3 DEFINE.SYSIN Sample input for the Communication File Restore Utility

1. Replace the two occurrences of 'inform.infcom' with the name of the communication file.
2. Replace ’volume’ with the volume serial number of the disk device for the communication file.
3. Specify 'nnn' as the number of blocks. The default is 409.
4. Replace ‘user’ with any valid characters as specified in the IDCAMS parameter definitions.
5. Replace 'mmm mmm' with the expected block size or control interval size. The default is 4088.

   You must also change the BLKSIZE entry in the QFILE Parameter of the PARMBLK if you need a different block size.

Note: For more information on how to calculate VSAM space (records) and code PARMBLK parameters, see the A*dvantge VISION:Inform Installation Guide for CICS*.

The number of blocks must correlate to parameter entries in PARMBLK.

Provide the IDCAMS statements to delete the old communication file and define the new communication file in DEFINE.SYSIN.

Generated Output

The Communication File Restore Utility, CMRESTOR, creates one informational listing, REST.INFPRINT.

This listing summarizes the results of executing the Communication File Restore Utility. It contains a list of the members that were restored, as well as the communication file space utilization statistics (used blocks, free blocks, and so on).

Space Utilization Statistics

Each time you run the VISION:Inform Communication File Restore Utility against the communication file, the utility generates space utilization statistics and writes data to the INFPRINT summary listing.

A sample of the actual message text follows. The asterisks in the message text indicate where the utility inserts numerical values for each run.

```
**S801 Of ******* overflow blocks, ******* are free, and ******* are in use.
**S802 Of ******* root blocks, ******* are free, and ******* are in use.
**S803 Of ******* root blocks, ******* have overflowed.
```

With these space utilization statistics, the system administrator can now determine:

- When to allocate additional space for the library (usually when the number of available overflow blocks becomes too small).
- When to increase the number of root blocks (usually when the number of available root blocks becomes too small).
- When to increase the block sizes (usually when the number of overflow blocks becomes too high).
- When to increase the bucket sizes (usually when the number of overflow blocks becomes too high).

Chapter 6

DB2 Quick Start Utility (DB2QS)

The DB2 Quick Start Utility, DB2QS, generates VISION:Inform file definitions from existing DB2 table (relational) definitions.

- During DB2 Quick Start Utility processing, each DB2 table becomes a separate segment in the VISION:Inform file definition that is being created.
- The DB2 Quick Start Utility retrieves DB2 column information from the DB2 SYSCOLUMNS table, converts it into VISION:Inform field specifications and adds it to the segment being created.
- The DB2 Quick Start Utility creates one VISION:Inform field statement (L0) for each column in the specified DB2 table.

You can generate multiple file definitions in a single execution of the DB2 Quick Start Utility. The utility stores each generated file definition as a separate member in the indicated source definition library.

The DB2 Quick Start Utility uses the specified file name as the member name when it saves the generated definition in the source definition library.

- If you specify an existing member name, the utility replaces the existing member when the new definition is saved.
- If you specify a new member name, then the utility creates a new member.

Once the utility creates a VISION:Inform file definition, you can modify file information with the Definition Processor.

Note: You can also run this utility interactively under TSO/ISPF using the Definition Processor Import option.

Utility Component

The DB2 Quick Start Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(DB2QS)

This sample JCL to execute the DB2 Quick Start Utility is in the VISION:Inform JCL library, member DB2QS.

Flow Diagram

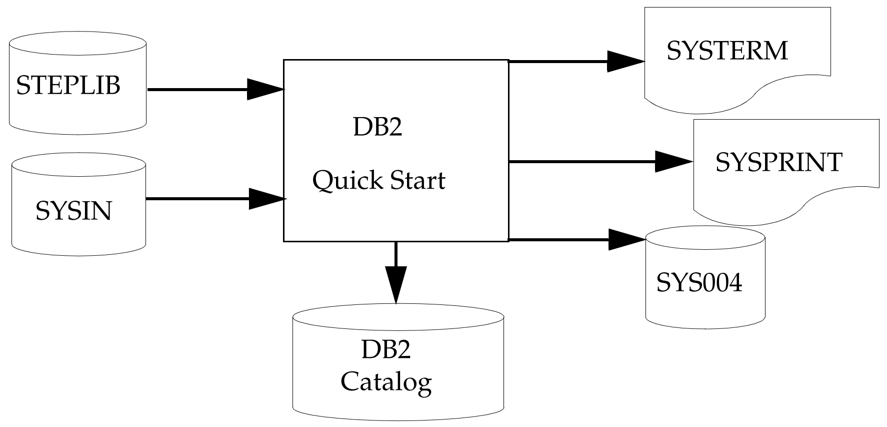

Figure 6-1 DB2 Quick Start Utility Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

The DB2 Quick Start Utility uses the following two input data sets:

STEPLIB	Specifies the VISION:Inform installation load library and the DB2 load library.
SYSIN	Specifies the appropriate DB2 Quick Start Utility control statements.

The DB2 Quick Start Utility produces two output files:

SYSTERM	Contains detailed descriptions of unexpected error conditions when coded as a SYSOUT file (SYSOUT=*).
SYSPRINT	Contains a summary report on the file definition generation process. Use the PRINT parameter on the SEGMENT control statement to control the contents of this report. For more information, see Specifying Control Statements.

SYS004	The utility writes the generated file definitions to the partitioned data set indicated by the DD name SYS004.

Executing the Utility

The DB2 Quick Start Utility is a batch utility. Before you execute this utility, use the DB2 Quick Start Data Base Request Module (DBRM) as input to a DB2 bind to create a DB2 plan for accessing the DB2 SYSCOLUMNS table.

- After you create the DB2 plan by binding the DB2 Quick Start Utility Data Base Request Module (DBRM), run the DB2 Quick Start Utility by submitting the JCL provided in INFORM.JCL in member DB2QS.

  This JCL contains an instream procedure followed by JCL statements to execute the procedure and sample DB2 Quick Start control statements.

- You can use the sample control statements to create a VISION:Inform file definition from the sample tables provided with DB2 (DSN8610.DEPT, DSN8610.EMP, and DSN8610.PROJ).

  If you have created a DB2 plan by binding the DB2 Quick Start DBRM, you can use these sample control statements to see how the DB2 Quick Start Utility works.

Note: The DBRM is on the installation tape. For information on binding the DB2 Quick Start Utility DBRM during installation, see the *Advantage VISION:Inform Installation Guide for CICS*.

Figure 6-2 shows the JCL contained in INFORM.JCL in the member DB2QS. Review the JCL carefully.

Note: You can also run this utility interactively under TSO/ISPF using the Definition Processor Import option.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply values for the procedure variables—for information, see Specifying Procedure Variables.
3. Provide a DD statement override for DB2QS.SYSIN. This data set contains the required the DB2 Quick Start Utility input control statements used to control the generation of VISION:Inform file definitions. For detailed information on these control statements, see Specifying Control Statements.
4. Make any additional modifications, as required, for your installation's standards.

Specifying the Execution JCL

The following figure shows the JCL to execute the DB2 Quick Start Utility.

```
//* MEMBER DB2QS
//*********************************************************************
//* EXECUTE THE DB2 QUICK START UTILITY.                              *
//*                                                                   *
//* THIS UTILITY MAY ALSO BE INVOKED INTERACTIVELY UNDER TSO/ISPF     *
//* USING THE DEFINITION PROCESSOR COMPONENT IMPORT FUNCTION.         *
//*********************************************************************
//DB2QS   PROC LOADLIB=,
//             DB2LOAD=,
//             DEFLIB=
//DB2QS   EXEC PGM=DB2QS,REGION=1024K
//STEPLIB  DD  DISP=SHR,DSN=&LOADLIB
//         DD  DISP=SHR,DSN=&DB2LOAD
//SYSTERM  DD  DUMMY
//SYSPRINT DD  SYSOUT=*,DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SYS004   DD  DISP=OLD,DSN=&DEFLIB
//SYSIN    DD  DUMMY
//        PEND
//*********************************************************************
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           *
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          *
//*                                                                   *
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                *
//*    DB2LOAD - THE DB2 SYSTEM LOAD LIBRARY NAME.                    *
//*    DEFLIB  - THE INFORM DEFINITION LIBRARY.                       *
//*                                                                   *
//*  YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE          *
//*  DB2QS.SYSIN DD OVERRIDE STMT.                                    *
//*********************************************************************
//QS      EXEC DB2QS,
//             LOADLIB='INFORM.LOADLIB',
//             DB2LOAD='DB2.DSNLOAD',
//             DEFLIB='INFORM.DEFLIB'
//DB2QS.SYSIN DD *
DB2CNTL  DB2PLAN=DB2QS,DB2SYS=DB2
FILEGEN NAME=DB2QS,BUFFSIZE=1024K
SEGMENT NAME=DEPT,NUMBER=10,LEVEL=1,TABLE=DEPT,CREATOR=DSN8230,
        PRINT=ALL
NEWPAGE
SEGMENT NAME=EMPLOYEE,NUMBER=20,LEVEL=2,TABLE=EMP,CREATOR=DSN8230,
        PRINT=ALL
NEWPAGE
SEGMENT NAME=PROJECT,NUMBER=30,LEVEL=2,TABLE=PROJ,CREATOR=DSN8230,
        PRINT=ALL
```

Figure 6-2 Sample Execution JCL for the DB2 Quick Start Utility, DB2QS

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing JCL procedure.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
DB2LOAD	Specify the name of the DB2 load library.
DEFLIB	Specify the source definition library to which the new file definition should be written. DEFLIB must be a partitioned data set.

Figure 6-3 JCL Procedure Variables for the DB2 Quick Start Utility

Specifying Control Statements

The DB2 Quick Start Utility uses the following control statements as input:

DB2CNTL	Information from the DB2CNTL statement establishes the proper DB2 environment.
FILEGEN	Provides the name for the file definition that is being generated. The utility uses the NAME parameter specified on the FILEGEN statement as both the file name and the member name when the generated file definition is saved.
SEGMENT	The SEGMENT statement defines file segments. Each segment corresponds to a DB2 table. The utility retrieves the field information for the specified DB2 table from the DB2 SYSCOLUMNS table, converts it to VISION:Inform field specifications, and uses it to create field definitions for the current segment.
NEWPAGE	The NEWPAGE statement triggers a page eject on the DB2 Quick Start Utility summary report.

The function and syntax of these statements are described in the following sections.

Coding Rules

Follow the coding rules below when writing the DB2 Quick Start Utility control statements:

- Each non-continued control statement must contain a control statement command that identifies the control statement type.
- Each control statement can contain keyword parameters.
- Specify keyword parameters in any order.
- Separate the parameters by commas.
- Do not use embedded blanks between parameters.
- Specify each parameter unless stated otherwise.
- You can place a comma following the last parameter on a statement to continue the statement on to the next line.
- You can place comments after the last parameter with an intervening blank.
- Blank lines within the SYSIN file are printed on the summary report output to SYSPRINT. You can use blank lines, along with the NEWPAGE control statement to format the DB2 Quick Start Utility summary report.
- The utility scans columns 1 through 71 of the control statement and ignores columns 72 through 80.

DB2CNTL Control Statement

DB2CNTL DB2PLAN=*planame*, DB2SYS=*sysname*

DB2CNTL	Required. DB2CNTL is a control statement command that identifies the control statement type. Use the DB2 control statement to establish the proper DB2 environment. The DB2 Quick Start Utility uses the TSO Attach Facility to connect to DB2. Static SQL is used to process information in the DB2 SYSCOLUMNS table.
DB2PLAN	Required. The DB2PLAN parameter specifies the name of the DB2 plan to be used when accessing DB2 tables. You create the DB2 plan during the installation process by binding the provided DB2 Quick Start Utility DBRM.
DB2SYS	Required. The DB2SYS parameter specifies the name of the DB2 system to be used.

FILEGEN Control Statement

FILEGEN NAME=*filedef*, BUFFSIZE=*nnnn*, FLDPREFIX=*pre*, FLDNAME=, HEADING=, LOGREL=, LONGNAME=, DESCRIPT=, DATEFLD=

The parameters are:

FILEGEN — Required. FILEGEN is a control statement command that identifies the control statement type. The FILEGEN control statement starts a new file definition.

NAME — Required. The NAME parameter specifies the name of the file definition that is being generated.

- Begin the name with an alphabetic letter.
- Specify the remaining characters as alphanumeric characters.
- Specify from 1 to 8 characters.

When the utility saves the generated file definition, it uses the file name as the member name.

- If you specify an existing member name, the utility overwrites the existing member.
- If you specify a new member name, the utility creates a new member.

BUFFSIZE — Optional. The BUFFSIZE parameter defines the size of the buffer needed to process a logical record.

- Enter a number from 1 to 32760.
- You can also enter multiples of 1024 as "nnnnk" where "nnnn" is a number between 1 to 9999.

FLDPREFX — Optional. The FLDPREFX parameter specifies the prefix for generating primary field names in the VISION:Inform file definition.

- Specify from 1 to 3 characters for the prefix. Make the first character of the prefix alphabetic. Primary field names are required, must be unique, and must contain from 1 to 8 characters.
- Since DB2 field names can be longer than eight characters and are not necessarily unique within the first eight characters, DB2 Quick Start automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number.
- If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format F*nnnnnnn* where *nnnnnnn* is a number from 0000001 to 9999999.

For more information, see Specifying VISION:Inform Field Names.

FLDNAME	Optional. The FLDNAME parameter specifies how the field name (short name) is derived. Enter GEN to specify that the name is to be generated using the FLDPREFIX parameter convention. Enter TRUNC to specify that the name is to be a truncation (first 8 characters) of the DB2 table column name. If you omit the FLDNAME parameter, the system assumes an entry of GEN. Note that an entry of TRUNC can result in duplicate names that you need to resolve by modifying the generated code.
HEADING	Optional. The HEADING parameter specifies whether or not the Column Heading specifications (Ln statements) in the generated definition should contain the DB2 Label information for the SYSCOLUMNS table or the DB2 column name. Enter LABELS if the label information is to be used or COLUMN if the column name is to be used. If you omit the HEADING parameter, the system assumes the LABEL specification.
LOGREL	Optional. The LOGREL parameter specifies whether or not the DB2 Quick Start utility should generate LR statements for the definition. Enter YES to cause LR statements to be generated or NO to suppress the generation of LR statements. If you omit the LOGREL parameter, the system assumes an entry of YES. You must specify the foreign keys for a table to DB2 for this function of the utility to work correctly.
LONGNAME	Optional. The LONGNAME parameter specifies whether or not the DB2 Quick Start utility should generate LX statements for the definition. An entry of YES specifies that LX statements are to be generated containing the full column name. An entry of NO specifies that LX statements should not be generated. If you omit the LONGNAME parameter, the system assumes an entry of YES.
DESCRIPT	Optional. The DESCRIPT parameters specifies whether or not the DB2 Quick Start utility should generate D1 statements for the definition. An entry of YES specifies that D1 statements are to be generated using the Label information in the SYSCOLUMNS table. An entry of NO specifies that D1 statements should not be generated. If you omit the DESCRIPT parameter, the system assumes an entry of YES.
DATEFLD	Optional. The DATEFLD parameter specifies how to handle DB2 columns with a data type of DATE. An entry of YES specifies that a DB2 data type of DATE should be defined as a Lilian Date (Type D) to VISION:Inform. An entry of NO specifies that a DB2 data type of DATE should be defined as a character string (Type C) with a length of 10 to VISION:Inform. If you omit the DATEFLD parameter, the system assumes an entry of NO.

SEGMENT Control Statement

SEGMENT NAME=*segname*, NUMBER=*nnn*, LEVEL=*n*, TABLE=*tablname*, CREATOR=*creatorid*, PRINT= {DB2 | ANSWER | ALL },

SEGMENT	Required. SEGMENT is a control statement command that identifies the control statement type. Each VISION:Inform segment corresponds to a specific DB2 table.
NAME	Required. The NAME parameter assigns a name to a segment. ■ Begin the name with an alphabetic letter. ■ Specify the remaining characters as alphanumeric characters. ■ Specify from 1 to 8 characters.
NUMBER	Required. The NUMBER parameter assigns a number that uniquely identifies the segment. ■ Enter a number from 1 to 255. ■ Make subordinate segments a number larger than the parent segment and smaller than any dependent segments.
LEVEL	Required. The LEVEL parameter specifies the subordination of segments. ■ Make the root segment level 1. ■ Assign all subordinate segments a number from 2 to 9.
TABLE	Required. The TABLE parameter specifies the name of the DB2 table containing column information to be used when generating the segment field definitions.
CREATOR	Required. The CREATOR parameter qualifies the table name. Enter the authorization or creator ID for the table, or enter an asterisk (*). If you enter an asterisk, field information is generated from all tables with the specified table name, regardless of the creator ID.
PRINT	Optional. The PRINT parameter controls the information that is printed on the summary report. ■ To print DB2 field information, enter DB2. ■ To print VISION:Inform field information, enter ANSWER. ■ To print both DB2 and VISION:Inform field information, enter ALL.

NEWPAGE Control Statement

NEWPAGE

NEWPAGE	Required. NEWPAGE is a control statement command that causes a page eject in the summary report that is written to SYSPRINT. There are no parameters for this command.

Editing Generated VISION:Inform File Definitions

After the DB2 Quick Start Utility generates a skeletal VISION:Inform file definition, you edit and validate the file definition using the Definition Processor prior to use. Consider the following items when editing the generated relational file definition:

Segment Key Assignment	Specify each segment with at least one field as the key. You can assign additional keys if needed.
Logical Relationships	Provide logical relationships for segments defined at levels 2 to 9.
Segment Information	Provide additional segment information such as segment (row) order and suppress duplication as needed.
Field Information	You can modify primary field name assignments. Provide additional field information such as rounding, editing, column headings, and automatic table lookup results, as needed.

Specifying VISION:Inform Field Names

Assign all fields within a VISION:Inform file definition a unique name containing from
1 to 8 characters. This name is referred to as the primary field name.

When you build a skeletal VISION:Inform file definition from a DB2 field definition, the DB2 Quick Start Utility assigns a unique primary field name to each field. The field name contains from 1 to 8 characters.

- Since DB2 field names can be longer than eight characters and are not necessarily unique within the first eight characters, the DB2 Quick Start Utility automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number.
- If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format Fnnnnnnn where nnnnnnn is a number from 0000001 to 9999999.

The DB2 Quick Start Utility uses the DB2 column name to generate the external column name specification.

Once the DB2 Quick Start Utility creates the skeletal file definition, you import the file definition into the Definition Processor and replace the generated primary field names with more descriptive primary field names.

Generating VISION:Inform Field Information

The following shows how the information in the DB2 SYSCOLUMNS column maps to the generated VISION:Inform field statements.

DB2 SYSCOLUMNS Field	VISION:Inform Field
NAME	Generates the Column Name specification.
COLTYPE	Generates the Field Type specification. See Figure 6-5 for more information.
LENGTH	Generates the Field Length specification. See Figure 6-5 for more information.
SCALE	Generates the Decimal Places specification. See Figure 6-5 for more information.
KEYSEQ	Generates the Segment Key Value specification.
LABEL	Generates the Column Heading specifications.

Figure 6-4 Generating VISION:Inform Field Information

The following table displays the conversion rules for field types and field lengths. The generated VISION:Inform field length for a GRAPHIC, VARGRAPHIC, and LONG VARGRAPHIC string is 2*n, where n is the DB2 length of the string (number of DBCS characters).

DB2 Field Type	DB2 Field Length	VISION:Inform Field Type	VISION:Inform Field Length
INTEGER	4	Fixed (F)	4
SMALLINT	2	Fixed (F)	2
FLOAT	4	Floating Point (E)	4
FLOAT	8	Floating Point (E)	4 (Truncate)
DECIMAL	Up to 31 digits	Packed Decimal (P)	Up to 15 bytes — possible truncation.
CHAR	Up to 254	Character (C)	Up to 255
VARCHAR	Up to maximum record size	Variable (V)	Up to 99H — possible truncation.
LONG VARCHAR	Up to maximum page size	Variable (V)	Up to 99H — possible truncation.
GRAPHIC	Up to 127	Character (C)	Up to 255
VARGRAPHIC	Up to maximum record size	Variable (V)	Up to 99H — possible truncation.
LONG VARGRAPHIC	Up to maximum page size	Variable (V)	Up to 99H — possible truncation.
DATE	4	Character (C)	10
TIME	3	Character (C)	8
TIMESTAMP	10	Character (C)	26

Figure 6-5 DB2 to VISION:Inform Field Conversions

Chapter 7

Definition Convert Utility (CNVRTDEF)

The Definition Convert Utility, CNVRTDEF, converts existing file definitions, logical data view definitions, table definitions, external requests, and ASL procedures into the format and order required for use with the Definition Processor component of VISION:Inform. This utility is for customers who are upgrading from *Answer/DB* to VISION:Inform.

Utility Component

The Definition Convert Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(CNVRTDEF)

This sample JCL to execute the Definition Convert Utility is in the VISION:Inform JCL library, member CNVRTDEF.

Input to the Definition Convert Utility is in the form of a sequential data set or PDS members. The utility can process multiple definitions in a single execution.

Output from the Definition Convert Utility is in the form of a single PDS member for each definition that has been converted.

Flow Diagram

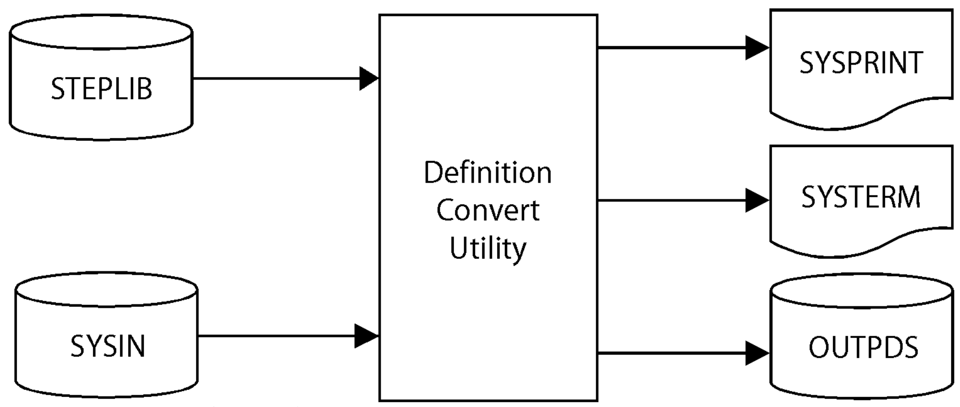

Figure 7-1 Definition Convert Utility Flow Utility Diagram

Note: Utility flow diagram names correspond to JCL DD names.

The Definition Convert Utility uses the following two input data sets:

STEPLIB	STEPLIB specifies the VISION:Inform installation load library.
SYSIN	SYSIN specifies an input file to be converted. You can specify SYSIN in any of the following ways:

- A sequential data set.
- A member in a partitioned data set (PDS).
- Multiple concatenated PDS members from the same PDS.
- As SYSIN instream data.

For details on concatenated input and instream SYSIN data, see Providing DD Statement Overrides.

You can specify multiple definitions of any type in the input.

The Definition Convert Utility processes:

- File definitions (FD)
- Logical data view definitions (DB)
- Table definitions (TB)
- External requests (ER)
- ASL procedures (ER)

The Definition Convert Utility produces the following three output data sets:

SYSPRINT	The SYSPRINT data set contains a summary report of the converted definitions. The utility lists statements that were discarded by member.
SYSTERM	This SYSOUT file contains detailed descriptions of unexpected error conditions.
OUTPDS	The OUTPDS DD name defines an output partitioned data set to which the Definition Convert Utility writes the converted definitions.

Executing the Utility

The Definition Convert Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(CNVRTDEF)

This JCL contains an instream procedure (PROC), followed by the steps to execute the procedure.

Specifying the Execution JCL

Figure 7-2 shows the sample JCL contained in INFORM.JCL in the member CNVRTDEF. Review the supplied JCL carefully.

At a minimum, make the following changes before submitting the JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the Definition Convert Utility.

```
//* MEMBER CNVRTDEF
//********************************************************************
//* THIS PROCEDURE RUNS THE DEFINITION CONVERT UTILITY IN ORDER TO   *
//* CONVERT THE FOLLOWING ITEMS FROM OLDER VISION:INFORM RELEASE     *
//* FORMATS TO THE DEFINITION PROCESSOR FORMAT:                      *
//*                                                                  *
//*         1) FILE DEFINITIONS              2) TABLE DEFINITIONS    *
//*         3) LOGICAL DATAVIEW DEFINITIONS  4) ASL PROCEDURES       *
//*         5) EXTERNAL REQUESTS                                     *
//********************************************************************
//CONVERT PROC LOADLIB=,
//            RGN=,
//            RUNMODE=,
//            OUTPDS=,
//            INPUT=
```

Figure 7-2 Sample Execution JCL for the Definition Convert Utility, CNVRTDEF

```
//CONVRT EXEC PGM=CONVERT,REGION=&RGN,PARM='&RUNMODE'                   00160001
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00170001
//SYSPRINT DD SYSOUT=*                                                  00180001
//SYSTERM  DD SYSOUT=*                                                  00190001
//OUTPDS   DD DISP=SHR,DSN=&OUTPDS                                      00200001
//SYSIN    DD DISP=SHR,DSN=&INPUT                                       00210001
//       PEND                                                           00220001
//********************************************************************* 00230000
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           * 00240001
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          * 00250001
//*                                                                   * 00260000
//*  LOADLIB - NAME OF THE INFORM INSTALLATION LOAD LIBRARY.          * 00270001
//*  RGN     - THE REGION SIZE. DEFAULT IS 512K.                      * 00280001
//*  RUNMODE - THE CONVERT UTILITY RUN MODE. SPECIFY EITHER 'TEST'    * 00290001
//*            OR 'UPDATE'. THE DEFAULT IS 'TEST'.                    * 00300001
//*  OUTPDS  - THE NAME OF THE PARTITIONED DATASET WHERE THE          * 00310001
//*            CONVERTED OUTPUT WILL BE STORED.                       * 00320001
//*  INPUT   - SPECIFY A SINGLE INPUT STREAM AS EITHER A SEQUENTIAL   * 00330001
//*            FILE OR A PDS MEMBER. EXAMPLE:                         * 00340001
//*            SEQUENTIAL  -- //  INPUT='USER.SEQ.FILEDEFS'           * 00350001
//*            PARTITIONED -- //  INPUT='USER.PDS.FILEDEFS(MEMBER)'   * 00360001
//********************************************************************* 00370000
//CONVERT EXEC CONVERT,                                                 00380001
//             LOADLIB='INFORM.LOADLIB',                                00390001
//             RGN=512K,                                                00400000
//             RUNMODE=TEST,                                            00410000
//             OUTPDS='USER.PDS.CNVDEFS',                               00420000
//             INPUT='USER.SEQ.FILEDEFS'                                00430000
```

Figure 7-2 Sample Execution JCL for the Definition Convert Utility, CNVRTDEF

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify to run the Definition Convert Utility JCL procedure, CNVRTDEF.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
RGN	The region size for execution of the Definition Convert Utility. The default is 512K.
RUNMODE	Specify either TEST or UPDATE. ■ The default, TEST, causes the Definition Convert Utility to execute without writing anything to the OUTPDS data set. ■ By specifying UPDATE, converted items are written to OUTPDS.
OUTPDS	This is the output data set for the converted items. ■ Specify an existing PDS (with no member name specified). ■ Specify this parameter even when RUNMODE=TEST.
INPUT	Specify the SYSIN input here in one of the following formats: ■ Sequential data set, as in: INPUT='user.seq.filedefs' ■ A PDS member, as in: INPUT='user.filedef.pds(mbrname)'

Figure 7-3 JCL Procedure Variables for the Definition Convert Utility

Providing DD Statement Overrides

Use DD overrides to change the specifications for the SYSIN data set. You can do this in one of two ways:

- To specify multiple PDS members from the same SYSIN PDS, specify the first PDS member in the INPUT procedure variable in the normal fashion for a PDS member. Then, add the following DD override after the instream procedure call:

```
//SYSIN    DD DISP=SHR,DSN=THE.SAME.PDSNAME(MEMBER)
//         DD DISP=SHR,DSN=THE.SAME.PDSNAME(MEMBER1)
      .
      .
      .
```

- To specify the SYSIN item to be converted as instream data, override SYSIN after the instream procedure call as follows:

```
//SYSIN    DD *
      .
      .
      .
input to be converted
      .
      .
      .
```

Operational Characteristics

The Definition Convert Utility operational characteristics include:

- Supported Statement Types.
- Single Member Versus Multiple Member Input.
- OUTPDS Member Naming Conventions.
- Data Changes in Converted OUTPDS Members.
- Specifying Run Mode.
- Generated Output.

Supported Statement Types

The following shows the supported statement types by request or definition types. When the Definition Convert Utility encounters any statement type that is not in the supported statements list for that type, the utility discards the statement and issues a message.

Type	Supported Statements
TB — Table Definition	TB, AA, TE
FD — File Definition	FD, AA, LS, LM, L0, LA, LX, Dn, Ln, LR For LS statements: Each segment, including virtual segments, must have a unique number before the definition can be properly converted. For L0 statements: If you have L0 statements without corresponding LS statements identifying the segment, you must insert the LS statements before the definition can be properly converted even if it is a flat record with only one segment.
DB — Logical Data View	DB, RP, AA, RF, LB, WH, CR
ER — External Request	ER, AA, TF, PR If you have requests that do not begin with ER statements, insert an ER statement in front of each request before the definition can be properly converted.
ER — ASL Procedure	ER, ##PROC, ##PEND, all statements between ##PROC and ##PEND

Figure 7-4 Definition Convert Utility Supported Statement Types

Single Member Versus Multiple Member Input

When the SYSIN file for the Definition Convert Utility specifies a single item containing a single Definition Processor object, the processing for that member is uncomplicated. When the Definition Convert Utility encounters an unsupported statement type, the utility discards the statement, issues a message, and places the remaining statements in the proper order for the Definition Processor.

When the SYSIN file contains multiple Definition Processor objects to be converted, use "Member Start" statements and "Member End" statements to control the conversion process.

“Member Start” Statements

A "Member Start" statement is one of the following:

- FD (file definition).
- DB (logical data view definition).
- TB (table definition).
- ER (external requests and ASL procedures).

Note: If you have requests that do not begin with ER statements, insert an ER statement in front of each request before the definition can be properly converted.

“Member End” Statements

A "Member End" statement is either one of the "Member Start" statements listed above or one of the following:

- RC (run control).
- AA (comment) — where the name in columns 1 – 8 of the AA statement does not match the name in columns 1 – 8 of the "Member Start" statement that defined the beginning of the item to be converted.
- EOF (end of file) on the SYSIN data set.

Processing” Member Start” — “Member End” Groups

The utility processes all statements from and including the "Member Start" statement, to, but not including, the "Member End" statement, as a single item to be converted.

The utility processes statements that occur between "Member End" and "Member Start" statements as part of the next item defined by a "Member Start" statement.

OUTPDS Member Naming Conventions

Normally, the member name of a converted item used in the OUTPDS data set is the name that appeared in columns 1 - 8 of the "Member Start" statement for that item.

However, prior to writing the member to OUTPDS, the Definition Convert Utility determines if that member name already exists in the OUTPDS data set. If a duplicate exists, the utility issues a message. If the name would constitute an invalid PDS member name, then the utility generates a member name. The naming convention for the generated member names is as follows:

- An 8-character name is always generated.
- The first two positions of the name are assigned as shown in the following:

Item Type	Positions 1-2 Of Name
File Definition	FD
Logical Data View Definition	DB
Table Definition	TB
External Request	ER
ASL Procedure	AS

Figure 7-5 Generated Names

- Positions 3 – 8 of the generated name contain a 6-digit unique number generated sequentially by type in the Definition Convert Utility.

Data Changes in Converted OUTPDS Members

Normally, the Definition Convert Utility does not change the text of statements written to the OUTPDS data set. However, the utility makes changes to statements in the converted items in two places:

- The DELETE column in DB statements (col. 19), FD statements (col. 19), and TB statements (col. 12) is always set to an R for replace.
- Column 72 of ER statements that begin ASL procedures is set to a P.

Specifying Run Mode

When you execute the Definition Convert Utility, you can set the run mode to TEST or UPDATE in the JCL procedure. The difference between the TEST and UPDATE modes is that in TEST mode, the OUTPDS data set is neither opened nor written to. The directory for the OUTPDS data set is always accessed in both run modes, to determine if a duplicate member name exists.

Generated Output

The Definition Convert Utility generates a SYSPRINT listing of members containing only one statement, if any such occurred. These items are usually caused by multiple FD, TB, ER, or DB statements with the same name in the same execution of the Definition Convert Utility. Normally, these can be deleted. However, the list of such items is provided so that you can retain the statements if necessary.

Chapter 8

Field Description Merge Utility (MERGHLP)

The Field Description Merge Utility, MERGHLP, merges existing *Answer*/DB field descriptions, known as FDHELPs, into existing VISION:Inform file definition source members. Once this is done, you can process and modify field descriptions as part of the file definition, rather than as a separate entity.

Note: Take the foreground library offline to run this utility.

This utility is for customers who are upgrading from *Answer*/DB to VISION:Inform. Run this utility if your prior release is *Answer*/DB Release 4.7 or earlier.

The Field Description Merge Utility:

- Retrieves existing field descriptions from the *Answer*/DB or VISION:Inform foreground library.
- Reformats the field descriptions into the new VISION:Inform format.
- Creates a new file definition that contains the merged file definition and field description source statements. The utility does not modify or change the original definition in any other way.

Once the utility has merged the existing field descriptions into the file definition, you can review and modify them, as part of the file definition, using the Definition Processor.

When you run the utility, the field description merge is done on a file-by-file basis, by specifying a particular file definition name.

Before using the Field Description Merge Utility, convert the prior release foreground library—for information, see the *Advantage VISION:Inform Getting Started.*

Throughout the rest of this chapter, the library that the existing field descriptions are being retrieved from is referred to as the VISION:Inform foreground library.

Utility Components

The Field Description Merge Utility components are:

Sample Execution JCL — INFORM.JCL(MERGHLP)

This sample JCL to execute the Field Description Merge Utility is in the VISION:Inform JCL library, member MERGHLP.

Source Library — INFORM.SRCLIB(MERGHLP)

The source library contains source program statements used by the Field Description Merge Utility.

Utility Library — INFORM.UTLIB

The utility library contains the data definitions that are needed by the Field Description Merge Utility.

Flow Diagram

The following diagram shows steps for the field description merge process.

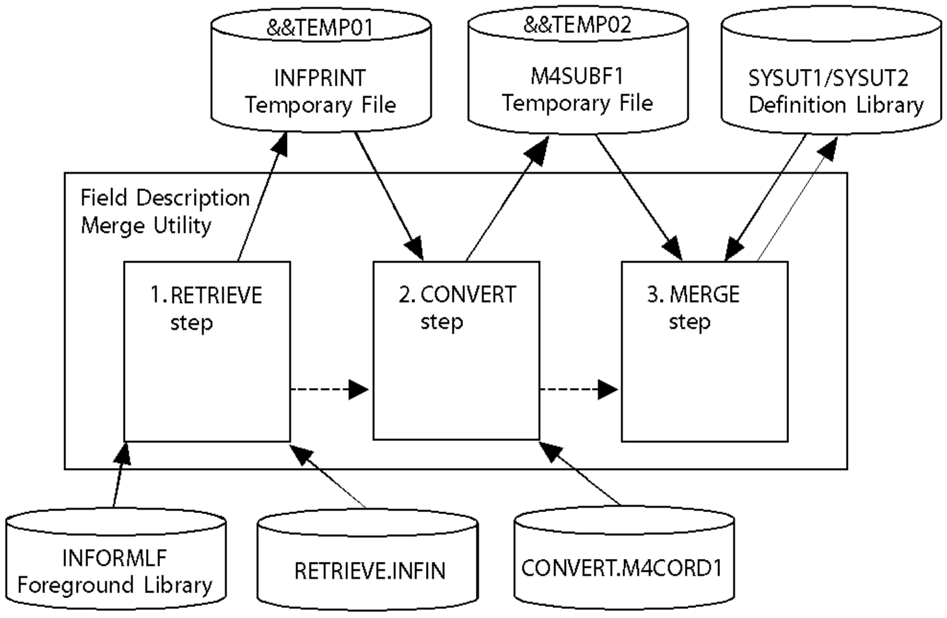

Figure 8-1 Field Description Merge Utility Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names and job step names.

Utility Flow

The utility steps are:

Note: Because the Field Description Merge Utility must access the VISION:Inform foreground library, take the foreground library offline.

1. RETREV step

   The utility retrieves all the existing field descriptions for a particular file definition from the specified VISION:Inform foreground library and stores them in a temporary disk data set (&&TEMP01). This temporary file is input to Step 2.

   The control statements specified in RETREV.INFIN determine which set of field descriptions is actually retrieved from the foreground library. Field descriptions are retrieved by the name of the specified file definition.

2. CONVRT step

   The utility converts the field descriptions to VISION:Inform format by reading the temporary file that was created in Step 1. You specify control statements in CONVRT.M4CORD1 with the name the file definition being processed. The utility reads, reformats, and writes field descriptions to a temporary disk data set (&&TEMP02). This temporary disk data set is input to Step 3.

   Step 2 also produces two reports providing summary information on the conversion process. The utility writes the reports to CONVRT.M4LIST.

Note: The step names correspond to JCL proc step names.

3. MERGE step

   The utility creates new file definition by merging the original file definition source with the converted field descriptions. The utility writes this new definition to the specified definition library as a library member. This new member contains the integrated file definition and field description specifications. The original file definition is not changed or modified in any other way.

Note: This step assumes that the original file definition source is available in the specified definition library.

Executing the Utility

The Field Description Merge Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(MERGHLP)

This JCL consists of an instream JCL procedure, followed by the steps to execute the procedure.

The Field Description Merge Utility merges field descriptions on a file-by-file basis. Each time you run the utility, it processes one file definition.

Specifying the Execution JCL

Figure 8-3 shows the sample JCL contained in INFORM.JCL in the member MERGHLP. Review this JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Complete the DD statement overrides for RETREV.INFIN and CONVRT.M4CORD1.
4. Make additional modifications, as needed, to conform to your installation's standards.

Note: Take the foreground library offline when you run the Field Description Merge Utility so that the utility can access the VISION:Inform foreground library.

The following figure shows the JCL to run the Field Description Merge Utility.

```
//* MEMBER MERGHLP
//*********************************************************************
//* EXECUTE THE FIELD DESCRIPTION MERGE UTILITY. RUN THIS UTILITY     *
//* ONLY AFTER CONVERTING THE OLD RELEASE'S FOREGROUND LIBRARY, AND   *
//* ONLY IF THE PREVIOUS RELEASE WAS ANSWER/DB 4.7 OR EARLIER.        *
//* NOTE - THIS SOURCE MEMBER SHOULD CONTAIN BLANKS IN COLUMNS 73-80,*
//*        AT LEAST IN ALL INLINE INPUT 'DD *' CONTROL STATEMENTS.    *
//*********************************************************************
//MERGHLP PROC LOADLIB=,
//             SORTLIB=,
//             UTILLIB=,
//             BGLIB=,
//             FGLIB=,
//             DEFLIB=,
//             OLDDEF=,
//             NEWDEF=,
//             SRCLIB=
//RETREV EXEC PGM=INFORMSB,REGION=2048K
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB
//SYSOUT   DD SYSOUT=*
//INFOPT   DD SYSOUT=*
//INFLIST  DD SYSOUT=*
//INFREPT  DD SYSOUT=*
//SORTLIB  DD DISP=SHR,DSN=&SORTLIB
//SORTWK01 DD SPACE=(CYL,5,,CONTIG),UNIT=SYSDA
//SORTWK02 DD SPACE=(CYL,5,,CONTIG),UNIT=SYSDA
//SORTWK03 DD SPACE=(CYL,5,,CONTIG),UNIT=SYSDA
//M4LIB    DD DISP=SHR,DSN=&BGLIB
//M4REPO   DD UNIT=SYSDA,SPACE=(TRK,(2,2))
//INFORMLF DD DISP=SHR,DSN=&FGLIB
//INFORMCF DD DUMMY
//INFPRINT DD DISP=(NEW,PASS,DELETE),DSN=&&TEMP01,
//             UNIT=SYSDA,SPACE=(TRK,(10,10)),
//             DCB=(DSORG=PS,RECFM=FBA,LRECL=133,BLKSIZE=1330)
//INFIN    DD DUMMY
//CONVRT EXEC PGM=MARKIV,REGION=2048K,COND=(0,NE)
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB
//SORTLIB  DD DISP=SHR,DSN=&SORTLIB
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//M4SORT   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//M4REPO   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//M4LIST   DD SYSOUT=*
//SYSOUT   DD DUMMY
//M4OLD    DD DISP=(OLD,DELETE,DELETE),DSN=&&TEMP01
//M4CORD1  DD DUMMY
//M4SUBF1  DD DISP=(NEW,PASS,DELETE),DSN=&&TEMP02,
//             UNIT=SYSDA,SPACE=(TRK,(10,10)),
//             DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=3120)
//M4SUBF2  DD DUMMY
//M4SUBF3  DD DUMMY
//M4SUBF4  DD DUMMY
//M4SUBF5  DD DUMMY
//M4SUBF6  DD DUMMY
//M4SUBF7  DD DUMMY
//M4SUBF8  DD DUMMY
//M4SUBF9  DD DUMMY
//M4LIB    DD DISP=SHR,DSN=&UTILLIB
//M4INPUT  DD DISP=SHR,DSN=&SRCLIB(MERGHLP)
//MERGE  EXEC PGM=IEBGENER,COND=(4,LT),REGION=2048K
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DISP=SHR,DSN=&DEFLIB(&OLDDEF)
//         DD DISP=(OLD,DELETE,DELETE),DSN=&&TEMP02
//SYSUT2   DD DISP=OLD,DSN=&DEFLIB(&NEWDEF)
```

Figure 8-2 Sample Execution JCL for the Field Description Merge Utility, MERGHLP (Page 1 of 2)

```
//       PEND
//*********************************************************************
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.            *
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                           *
//*                                                                    *
//*  LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                   *
//*  SORTLIB - THE SYSTEM SORT LOAD LIBRARY.                           *
//*  UTILLIB - THE INFORM UTILITY LIBRARY.                             *
//*  BGLIB   - THE INFORM BACKGROUND LIBRARY.                          *
//*  FGLIB   - THE INFORM FOREGROUND LIBRARY.                          *
//*  DEFLIB  - THE INFORM DEFINITION LIBRARY.                          *
//*  OLDDEF  - THE NAME OF THE EXISTING FILE DEFINITION.               *
//*  NEWDEF  - THE NAME OF THE NEW FILE DEFINITION BEING CREATED.      *
//*  SRCLIB  - THE INFORM SOURCE LIBRARY.                              *
//*********************************************************************
//MERGE  EXEC MERGHLP,
//            LOADLIB='INFORM.LOADLIB',
//            SORTLIB='SYS1.SORTLIB',
//            UTILLIB='INFORM.UTLIB',
//            BGLIB='INFORM.BGLIB',
//            FGLIB='INFORM.FGLIB',
//            DEFLIB='INFORM.DEFLIB',
//            OLDDEF='OLDNAME',
//            NEWDEF='NEWNAME',
//            SRCLIB='INFORM.SRCLIB'
//RETREV.INFIN DD *
 LOGON USER PSWD
 LIST ITEM OLDNAME TYPE FDHELP
 QUIT
//CONVRT.M4CORD1 DD *
OLDNAME
```

Figure 8-2 Sample Execution JCL for the Field Description Merge Utility, MERGHLP (Page 2 of 2)

Specifying Procedure Variables

The following table summarizes the procedure variables you specify when executing the Field Description Merge Utility JCL procedure, MERGHLP.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
SORTLIB	Specify the name of your system sort library.
UTLIB	Specify the name of the installed VISION:Inform utility library.
SRCLIB	Specify the name of the installed VISION:Inform source library. .
BGLIB	Specify the name of the installed VISION:Inform background library.
FGLIB	Specify the name of the installed VISION:Inform foreground library. The field descriptions are retrieved from this library.
DEFLIB	Specify the name of your VISION:Inform definition library. The current file definition is read from this library. The newly integrated definition is written to this library as a new member.
OLDDEF	Specify the name of the existing file definition.
NEWDEF	Specify a name for the new file definition. ■ If you specify an existing member, the utility replaces it. ■ If you specify a new member name, the utility creates new members.

Figure 8-3 JCL Procedure Variables for the Field Description Utility

Providing DD Statement Overrides

Provide the following DD statement overrides when executing the supplied JCL procedure for the Field Description Merge Utility.

Note: In the following examples, you modify the text in lowercase letters to meet your requirements.

Do not change the text in uppercase letters.

//RETREV.INFIN

This information controls the retrieval of the existing field descriptions from the specified VISION:Inform foreground library.

Replace ‘user’ and ‘pswd’ with the appropriate foreground library information.

Replace ‘oldname’ with the name of the file definition that you want to process. Match this name with the name specified in CONVRT.M4CORD1 and with the name specified for the OLDDEF variable.

```
//RETREV.INFIN  DD  *
  LOGON  user  pswd
  LIST ITEM  oldname  TYPE  FDHELP
  QUIT
/*
```

Figure 8-4 RETREV.INFIN Sample Input

//CONVRT.M4CORD1

Replace ‘oldname’ with the name of the file definition that you want to process. Match the name with the name specified on the RETREV.INFIN LIST statement and with the name specified for the OLDDEF variable in the MERGE step.

```
//CONVRT.M4CORD1  DD  *
  oldname
/*
```

Figure 8-5 CONVRT.M4CORD1 Sample Input

Generated Output

The Field Description Merge Utility produces the following informational reports summarizing the results of the integration process:

- List of statements converted.
- Summary of files processed.

The CONVRT step generates these reports and writes them to M4LIST.

Chapter 9

Glossary Utility (GLOSSARY)

The Glossary Utility, GLOSSARY, generates VISION:Inform hard copy source listings, also known as glossaries. The utility creates glossaries from information in the VISION:Inform background library. The Glossary Utility can generate glossaries for VISION:Inform table definitions, file definitions, and logical data view definitions. Glossaries are not available for requests or procedures.

Utility Component

The Glossary Utility is made up of the following component:

Sample Execution JCL — INFORM.JCL(GLOSSARY)

This sample JCL to execute the Glossary Utility is in the VISION:Inform JCL library, member GLOSSARY.

Flow Diagram

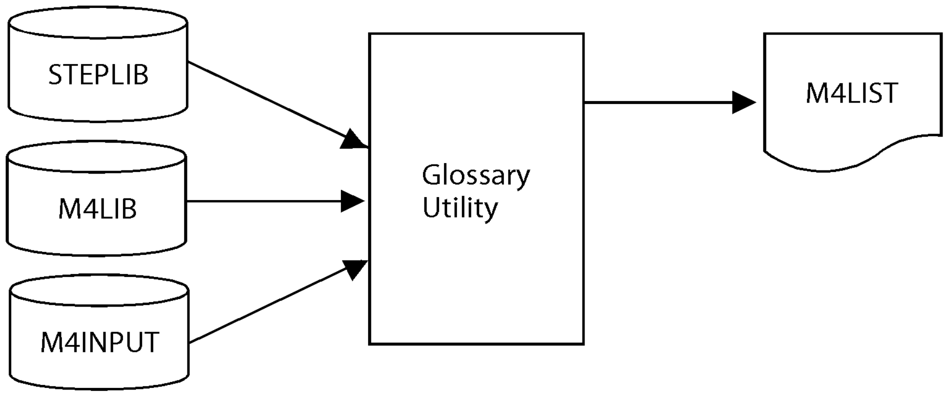

Figure 9-1 Glossary Utility Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

The Glossary Utility uses the following three input data sets:

STEPLIB	STEPLIB specifies the VISION:Inform installation load library.
M4LIB	M4LIB specifies the VISION:Inform background library from which the source will be retrieved.
M4INPUT	M4INPUT specifies the appropriate input control statements.

The Glossary Utility creates the following output file:

M4LIST	M4LIST specifies a SYSOUT file. The utility writes the glossary listing to this file.

Note: If the IBM Language Environment (LE) run-time library is not automatically available to this program, concatenate it to the STEPLIB DD statement.

Executing the Utility

The Glossary Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(GLOSSARY)

This JCL contains an instream procedure, followed by the steps to execute the procedure.

Specifying the Execution JCL

Figure 9-2 shows the sample JCL contained in INFORM.JCL in the member GLOSSARY. Review the JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Complete the DD statement override for GLOSSARY.M4INPUT by supplying the appropriate GLOSSARY.M4INPUT control statements.
4. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the Glossary Utility.

```
//* MEMBER GLOSSARY                                                     00010000
//********************************************************************* 00020003
//* THIS PROCEDURE WILL CREATE A HARD COPY GLOSSARY LISTING OF        * 00030003
//* SPECIFIED DATAVIEWS.                                              * 00040003
//* *** NOTE ***  IF THE LANGUAGE ENVIRONMENT LOAD LIBRARY IS NOT IN  * 00050003
//*               THE STANDARD LOAD LIBRARY LIST MADE AVAILABLE TO    * 00060003
//*               ALL BATCH PROGRAMS ON YOUR HOST SYSTEM, IT MUST BE  * 00070003
//*               CONCATENATED TO THE GLOSS.STEPLIB DD STATEMENT.     * 00080003
//********************************************************************* 00090003
//GLOSSRY PROC LOADLIB=,                                                00100000
//           BGLIB=                                                     00110001
//GLOSS  EXEC PGM=MARKIV,REGION=2048K                                   00120001
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00130002
//M4LIST   DD SYSOUT=*                                                  00140001
//M4LIB    DD DISP=SHR,DSN=&BGLIB                                       00150001
//M4INPUT  DD DUMMY                                                     00160001
//       PEND                                                           00170001
//********************************************************************* 00180003
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           * 00190003
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          * 00200003
//*                                                                   * 00210003
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                * 00220003
//*    BGLIB   - THE INFORM BACKGROUND LIBRARY.                       * 00230003
//*                                                                   * 00240003
//*  YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE          * 00250003
//*  GLOSS.M4INPUT DD OVERRIDE STMT. REPLACE THE "?" CHARACTER IN     * 00260004
//*  THE SAMPLE FD STATEMENT WITH AN "X" FOR A GLOSSARY SORTED BY     * 00270004
//*  FIELD LOCATION, OR A "Y" FOR A GLOSSARY SORTED BY FIELD NAME.    * 00280004
//********************************************************************* 00290003
//GLOSS  EXEC GLOSSRY,                                                  00300000
//           LOADLIB='INFORM.LOADLIB',                                  00310001
//           BGLIB='INFORM.BGLIB'                                       00320001
//GLOSS.M4INPUT DD *                                                    00330003
GLOSSARYRC                    B                                         00340000
FILENAMEFD         ?                                                    00350000
```

Figure 9-2   Sample Execution JCL for the Glossary Utility, GLOSSARY

Note: The term “data view” appears as one word in the product.

Specifying Procedure Variables

The following summarizes the procedure variables that you need to specify when executing the Glossary Utility JCL procedure, GLOSSARY.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
BGLIB	Specify the name of the VISION:Inform background library from which the glossary is to be generated.

Figure 9-3   JCL Procedure Variables for the Glossary Utility

Providing DD Statement Overrides

Provide the following required DD statement overrides when executing the supplied Glossary Utility. The following sections show samples for table definition glossaries, file definition glossaries, and logical data view glossaries.

Note: In the following examples, you modify the text shown in lowercase letters.

Do not change the text in uppercase letters.

//GLOSSARY.M4INPUT for Table Definition Glossaries

Specify glossary control statements in the M4INPUT file.

1. Replace the '?' with the following glossary format code.

   To create a standard table glossary, specify a glossary format code of '1' in column 13.
2. Replace 'tbname' with the name of the definition for which you want to produce a glossary. Specify the definition name in columns 1 - 8.

```
....+....+....+....+....+....+....+....+....+
//GLOSSARY.M4INPUT DD  *
GLOSSARYRC                        A      S
tbname   TB   ?
```

Figure 1 GLOSSARY.M4INPUT Sample Input for Table Definitions

//GLOSSARY.M4INPUT for File Definition Glossaries

Specify glossary control statements in the M4INPUT file.

1. Replace the '?' with one of the following glossary format codes.

A	Create alphabetical listing by field name within each segment.
B	Create sequential listing by location of fields within each segment.
1	Create abbreviated listing, alphabetical by field name within each segment.
D	Used with GDBI (Generalized Data Base Interface) file definitions to list the called mapping procedures.

   Specify the glossary format code in column 20.
2. Replace 'filename' with the name of the definition for which you want to produce a glossary. Specify the definition name in columns 1 - 8.

The following figure shows GLOSSARY.M4INPUT sample input:

```
....+....+....+....+....+....+....+....+....+
//GLOSSARY.M4INPUT DD  *
GLOSSARYRC                       A      S
filenameFD         ?
```

Figure 9-4 GLOSSARY.M4INPUT Sample Input for File Definitions

//GLOSSARY.M4INPUT for Logical Data View Definition Glossaries

Specify the glossary control statements in The M4INPUT file.

1. Replace the '?' with one of the following glossary format codes.

A	Create alphabetical listing by field name within each segment.
B	Create sequential listing by location of fields within each segment.
1	Create abbreviated listing, alphabetical by field name within each segment.

   Specify the glossary format code in column 20.

2. Replace 'ldvname' with the name of the definition for which you want to produce a glossary. Specify the definition name in columns 1 - 8.

The following figure shows GLOSSARY.M4INPUT sample input:

```
....+....+....+....+....+....+....+....+....+
//GLOSSARY.M4INPUT DD  *
GLOSSARYRC                    A      S
ldvname DB         ?
```

Figure 9-5 GLOSSARY.M4INPUT Sample Input for Logical Data View Definitions

Generated Output

The Glossary Utility writes the glossary listing to the M4LIST file.

Chapter 10

Initialization Utility (INIT)

The Initialization Utility, INIT, defines and initializes the foreground library, background library, and the communication file. The Initialization Utility uses IDCAMS utility statements to specify the VSAM cluster definitions.

Utility Component

The Initialization Utility is composed of the following components:

Sample Execution JCL — INFORM.JCL(INIT)

This sample JCL, which executes the Initialization Utility, is in the VISION:Inform JCL library, member INIT. This version of INIT defines a VISION:Inform foreground library that can be available for update by only one application at a time (VSAM SHAREOPTIONS (2, 3)).

Sample Execution JCL — INFORM.JCL(INIT2)

This sample JCL, which executes the Initialization Utility, is in the VISION:Inform JCL library, member INIT2. This version of INIT defines a VISION:Inform foreground library that can be available for update by more than one application at a time (VSAM SHAREOPTIONS (3,3)).

Flow Diagram

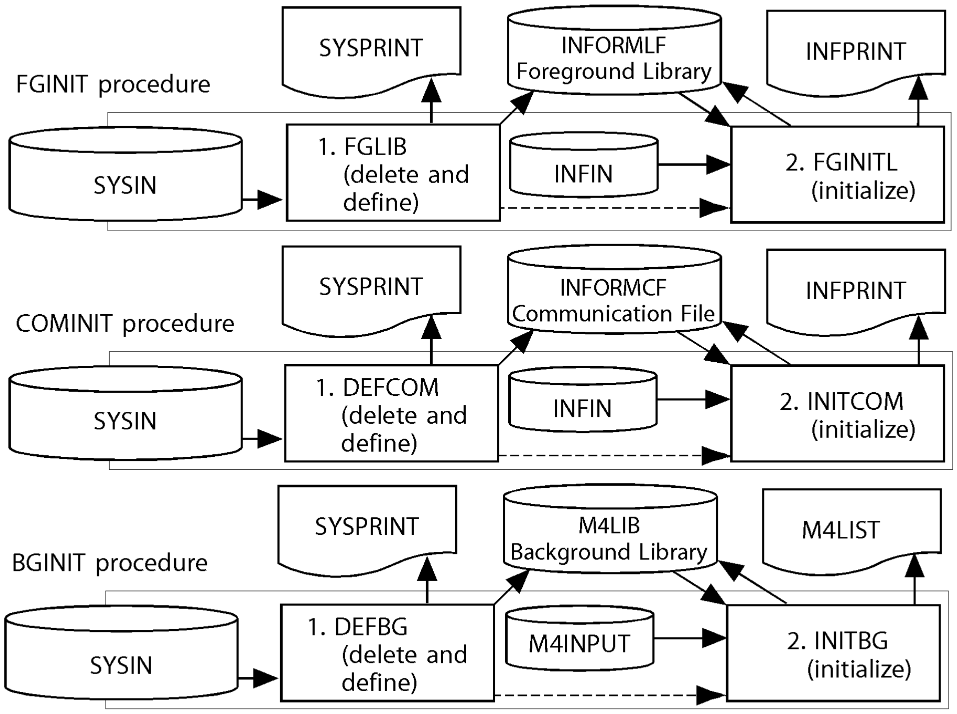

Figure 10-1 Initialization Utility Flow Diagram

Utility Flow

The Initialization Utility contains three procedures FGINIT, COMINIT, and BGINIT.

FGINIT Procedure

The FGINIT procedure consists of the following steps:

1. FGLIB step

   This step executes the VSAM IDCAMS utility to delete the old foreground library and define the new foreground library.

   The utility creates the SYSOUT data set, FGLIB.SYSPRINT.

   The acceptable return code for this step is 0.

2. FGINITL step

   This step initializes the foreground library.

   The procedure creates the SYSOUT data set, FGINITL.INFPRINT.

   The acceptable return code for this step is 0.

Note: The step names correspond to JCL job step names.

COMINIT Procedure

The COMINIT procedure consists of the following steps:

1. DEFCOM step

   This step executes the VSAM IDCAMS utility to delete the old and define the new communication file.

   The procedure creates the SYSOUT data set, DEFCOM.SYSPRINT.

   The acceptable return code for this step is 0.

2. INITCOM step

   This step initializes the communication file.

   The procedure creates the SYSOUT data set, INITCOM.INFPRINT.

   The acceptable return code for this step is 0.

BGINIT Procedure

The BGINIT procedure consists of the following steps:

1. DEFBG step

   This step executes the VSAM IDCAMS utility to delete and define the background library.

   The utility creates the SYSOUT data set, DEFBG.SYSPRINT.

   The acceptable return code for this step is 0.

2. INITBG step

   This step initializes the background library.

   The procedure creates the SYSOUT data set, INITBG.M4LIST.

   The acceptable return code for this step is 0.

Executing the Utility

The Initialization Utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(INIT) or INFORM.JCL(INIT2)

This JCL contains an instream procedure, followed by the steps to execute this procedure.

Note: For more information about jobs INIT and INIT2, see the *Advantage VISION:Inform Installation Guide for CICS.*

Specifying the Execution JCL

Figure 10-2 shows the sample JCL contained in INFORM.JCL in the member INIT. Review the JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL (member INIT in INFORM.JCL) to run the Initialization Utility, with the foreground library defined with VSAM SHAREOPTIONS (2,3).

```
//* MEMBER INIT                                                         00010000
//********************************************************************* 00020000
//* THIS JOB CONTAINS THREE PROCEDURES TO DEFINE AND INITIALIZE THE   * 00030000
//* FOREGROUND AND BACKGROUND LIBRARIES, AND THE COMMUNICATIONS FILE. * 00040000
//********************************************************************* 00050000
//FGINIT  PROC LOADLIB=,                                                00060000
//             FGLIB=                                                   00070000
//FGLIB   EXEC PGM=IDCAMS,REGION=512K                                   00080001
//SYSPRINT  DD SYSOUT=*                                                 00090000
//FGINITL EXEC PGM=INFORMUI,REGION=512K                                 00100001
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00110000
//INFPRINT  DD SYSOUT=*                                                 00120000
//INFORMLF  DD DISP=SHR,DSN=&FGLIB                                      00130000
//        PEND                                                          00140000
//COMINIT PROC LOADLIB=,                                                00150000
//             INFCOM=                                                  00160000
//DEFCOM  EXEC PGM=IDCAMS,REGION=512K                                   00170000
//SYSPRINT  DD SYSOUT=*                                                 00180000
//INITCOM EXEC PGM=INFORMUI,REGION=512K                                 00190000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00200000
//INFPRINT  DD SYSOUT=*                                                 00210000
//INFORMCF  DD DISP=SHR,DSN=&INFCOM                                     00220000
//        PEND                                                          00230000
//BGINIT  PROC LOADLIB=,                                                00240000
//             BGLIB=                                                   00250000
//DEFBG   EXEC PGM=IDCAMS,REGION=512K                                   00260000
//SYSPRINT  DD SYSOUT=*                                                 00270000
//INITBG  EXEC PGM=MARKUTIL,REGION=512K                                 00280000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00290000
//M4LIST    DD SYSOUT=*                                                 00300000
//M4WORK    DD DUMMY                                                    00310000
//M4LIB     DD DISP=OLD,DSN=&BGLIB                                      00320000
//        PEND                                                          00330000
//********************************************************************* 00340000
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THESE PROCEDURES.         * 00350001
//*  BEFORE YOU RUN THESE PROCEDURES, SPECIFY:                        * 00360001
//*                                                                   * 00370000
//*  FGLIB   - THE INFORM FOREGROUND LIBRARY.                         * 00380001
//*  INFCOM  - THE INFORM COMMUNICATION FILE.                         * 00390001
//*  BGLIB   - THE INFORM BACKGROUND LIBRARY.                         * 00400001
//*  LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                  * 00410001
//********************************************************************* 00420000
//*  FOR FGINIT,  ALSO CHANGE THE FGLIB.SYSIN STATEMENTS:             * 00430000
//*                                                                   * 00440000
//*      NAME('INFORM.FGLIB')                                         * 00450001
//*      NAME('INFORM.FGLIB.DATA')                                    * 00460001
//*      VOL('VOLUME')                                                * 00470000
```

Figure 10-2 Sample Execution JCL for the Initialization Utility, INIT (Page 1 of 3)

```
//*      OWNER('USER')                                                * 00480000
//*                                                                   * 00490000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION   * 00500000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                               * 00510000
//*                                                                   * 00520000
//*  *** DO NOT MAKE ANY CHANGES IN FGINITL.INFIN ***                 * 00530001
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***       * 00540000
//********************************************************************* 00550000
//FGINIT  EXEC FGINIT,                                                  00560000
//             LOADLIB='INFORM.LOADLIB',                                00570001
//             FGLIB='INFORM.FGLIB'                                     00580001
//FGLIB.SYSIN DD *                                                      00590000
      DELETE ('INFORM.FGLIB') CLUSTER PURGE                             00600001
      DEFINE CLUSTER -                                                  00610000
               (NAME('INFORM.FGLIB') -                                  00620001
               VOL('VOLUME') -                                          00630000
               RECORDS(385) -                                           00640000
               SHAREOPTIONS(2 3) -                                      00650000
               WRITECHECK NUMBERED -                                    00660000
               OWNER('USER')) -                                         00670000
             DATA(NAME('INFORM.FGLIB.DATA') -                           00680001
               CONTROLINTERVALSIZE(4096) -                              00690000
               RECORDSIZE(4088 4088))                                   00700000
//FGINITL.INFIN DD *                                                    00710001
ULSLIB                                                                  00720000
//********************************************************************* 00730000
//*  FOR COMINIT, ALSO CHANGE THE DEFCOM.INFIN STATEMENTS:            * 00740000
//*                                                                   * 00750000
//*      NAME('INFORM.INFCOM')                                        * 00760001
//*      NAME('INFORM.INFCOM.DATA')                                   * 00770001
//*      VOL('VOLUME')                                                * 00780000
//*      OWNER('USER')                                                * 00790000
//*                                                                   * 00800000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION   * 00810000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                               * 00820000
//*                                                                   * 00830000
//*  *** DO NOT MAKE ANY CHANGES IN INITCOM.INFIN ***                 * 00840000
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***       * 00850000
//********************************************************************* 00860000
//COMINIT EXEC COMINIT,                                                 00870000
//             LOADLIB='INFORM.LOADLIB',                                00880001
//             INFCOM='INFORM.INFCOM'                                   00890001
//DEFCOM.SYSIN DD *                                                     00900000
      DELETE ('INFORM.INFCOM') CLUSTER PURGE                            00910001
      DEFINE CLUSTER -                                                  00920000
               (NAME('INFORM.INFCOM') -                                 00930001
               VOL('VOLUME') -                                          00940000
               RECORDS(409) -                                           00950000
               SHAREOPTIONS(3 3) -                                      00960000
               WRITECHECK NUMBERED -                                    00970000
               OWNER('USER')) -                                         00980000
             DATA(NAME('INFORM.INFCOM.DATA') -                          00990001
               CONTROLINTERVALSIZE(4096) -                              01000000
               RECORDSIZE(4088 4088) -                                  01010000
               BUFFERSPACE(8192))                                       01020000
//INITCOM.INFIN DD *                                                    01030000
ULSCOM                                                                  01040000
//********************************************************************* 01050000
//*  FOR BGINIT, ALSO CHANGE THE DEFBG.SYSIN STATEMENTS:              * 01060000
//*                                                                   * 01070000
//*      NAME('INFORM.BGLIB')                                         * 01080001
//*      NAME('INFORM.BGLIB.DATA')                                    * 01090001
//*      VOL('VOLUME')                                                * 01100000
//*      OWNER('USER')                                                * 01110000
//*                                                                   * 01120000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION   * 01130000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                               * 01140000
//*                                                                   * 01150000
//*  *** DO NOT MAKE ANY CHANGES IN INITBG.M4INPUT ***                * 01160000
```

Figure 10-2   Sample Execution JCL for the Initialization Utility, INIT (Page 2 of 3)

```
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***       * 01170000
//********************************************************************* 01180000
//BGINIT  EXEC BGINIT,                                                  01190000
//             LOADLIB='INFORM.LOADLIB',                                01200001
//             BGLIB='INFORM.BGLIB'                                     01210001
//DEFBG.SYSIN DD *                                                      01220000
      DELETE  'INFORM.BGLIB' CLUSTER PURGE                              01230001
      DEFINE CLUSTER -                                                  01240000
               (NAME('INFORM.BGLIB') -                                  01250001
               VOL('VOLUME') -                                          01260000
               CYLINDERS(1 1) -                                         01270000
               SHAREOPTIONS(3 3) -                                      01280000
               CONTROLINTERVALSIZE(4096) -                              01290000
               RECORDSIZE(507 507) -                                    01300000
               WRITECHECK NUMBERED -                                    01310000
               OWNER('USER')) -                                         01320000
             DATA(NAME('INFORM.BGLIB.DATA'))                            01330001
//INITBG.M4INPUT DD *                                                   01340000
        UCINIT                                                          01350000
```

Figure 10-2   Sample Execution JCL for the Initialization Utility, INIT (Page 3 of 3)

The following figure shows the initialization JCL (member INIT2 in INFORM.JCL) which defines the foreground library with VSAM SHAREOPTIONS (3,3).

```
//* MEMBER INIT2                                                        00010000
//********************************************************************* 00020000
//* THIS JOB CONTAINS THREE PROCEDURES TO DEFINE AND INITIALIZE THE   * 00030000
//* FOREGROUND AND BACKGROUND LIBRARIES, AND THE COMMUNICATIONS FILE. * 00040000
//* THIS JOB SHOULD BE USED IF YOU ARE USING THE HOSTCONNECT SERVER   * 00050000
//* CLIENT OR IF YOU DESIRE TO ESTABLISH CONCURRENT UPDATE ACCESS TO  * 00060000
//* THE FOREGROUND LIBRARY.                                           * 00070000
//********************************************************************* 00080000
//FGINIT  PROC LOADLIB=,                                                00090000
//             FGLIB=                                                   00100000
//FGLIB   EXEC PGM=IDCAMS,REGION=512K                                   00110000
//SYSPRINT  DD SYSOUT=*                                                 00120000
//FGINITL EXEC PGM=INFORMUI,REGION=512K                                 00130000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00140000
//INFPRINT  DD SYSOUT=*                                                 00150000
//INFORMLF  DD DISP=SHR,DSN=&FGLIB                                      00160000
//        PEND                                                          00170000
//COMINIT PROC LOADLIB=,                                                00180000
//             INFCOM=                                                  00190000
//DEFCOM  EXEC PGM=IDCAMS,REGION=512K                                   00200000
//SYSPRINT  DD SYSOUT=*                                                 00210000
//INITCOM EXEC PGM=INFORMUI,REGION=512K                                 00220000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00230000
//INFPRINT  DD SYSOUT=*                                                 00240000
//INFORMCF  DD DISP=SHR,DSN=&INFCOM                                     00250000
//        PEND                                                          00260000
//BGINIT  PROC LOADLIB=,                                                00270000
//             BGLIB=                                                   00280000
//DEFBG   EXEC PGM=IDCAMS,REGION=512K                                   00290000
//SYSPRINT  DD SYSOUT=*                                                 00300000
//INITBG  EXEC PGM=MARKUTIL,REGION=512K                                 00310000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00320000
//M4LIST    DD SYSOUT=*                                                 00330000
//M4WORK    DD DUMMY                                                    00340000
//M4LIB     DD DISP=OLD,DSN=&BGLIB                                      00350000
//        PEND                                                          00360000
//********************************************************************* 00370000
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THESE PROCEDURES.         * 00380000
//*  BEFORE YOU RUN THESE PROCEDURES, SPECIFY:                        * 00390000
//*                                                                   * 00400000
//*  FGLIB   - THE INFORM FOREGROUND LIBRARY.                         * 00410000
//*  INFCOM  - THE INFORM COMMUNICATION FILE.                         * 00420000
```

Figure 10-3   Sample Execution JCL for the Initialization Utility, INIT2 (Page 1 of 3)

```
//*  BGLIB   - THE INFORM BACKGROUND LIBRARY.                         * 00430000
//*  LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                  * 00440000
//********************************************************************* 00450000
//*  FOR FGINIT,  ALSO CHANGE THE FGLIB.SYSIN STATEMENTS:             * 00460000
//*                                                                   * 00470000
//*      NAME('INFORM.FGLIB')                                         * 00480000
//*      NAME('INFORM.FGLIB.DATA')                                    * 00490000
//*      VOL('VOLUME')                                                * 00500000
//*      OWNER('USER')                                                * 00510000
//*                                                                   * 00520000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION   * 00530000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                               * 00540000
//*                                                                   * 00550000
//*  *** DO NOT MAKE ANY CHANGES IN FGINITL.INFIN ***                 * 00560000
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***       * 00570000
//********************************************************************* 00580000
//FGINIT  EXEC FGINIT,                                                  00590000
//             LOADLIB='INFORM.LOADLIB',                                00600000
//             FGLIB='INFORM.FGLIB'                                     00610000
//FGLIB.SYSIN DD *                                                      00620000
      DELETE ('INFORM.FGLIB') CLUSTER PURGE                             00630000
      DEFINE CLUSTER -                                                  00640000
               (NAME('INFORM.FGLIB') -                                  00650000
               VOL('VOLUME') -                                          00660000
               RECORDS(385) -                                           00670000
               SHAREOPTIONS(3 3) -                                      00690000
               WRITECHECK NUMBERED -                                    00700000
               OWNER('USER')) -                                         00710000
             DATA(NAME('INFORM.FGLIB.DATA') -                           00720000
               CONTROLINTERVALSIZE(4096) -                              00730000
               RECORDSIZE(4088 4088) -                                  00740001
               BUFFERSPACE(8192))                                       00741001
//FGINITL.INFIN DD *                                                    00750000
ULSLIB                                                                  00760000
//********************************************************************* 00770000
//*  FOR COMINIT, ALSO CHANGE THE DEFCOM.INFIN STATEMENTS:            * 00780000
//*                                                                   * 00790000
//*      NAME('INFORM.INFCOM')                                        * 00800000
//*      NAME('INFORM.INFCOM.DATA')                                   * 00810000
//*      VOL('VOLUME')                                                * 00820000
//*      OWNER('USER')                                                * 00830000
//*                                                                   * 00840000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION   * 00850000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                               * 00860000
//*                                                                   * 00870000
//*  *** DO NOT MAKE ANY CHANGES IN INITCOM.INFIN ***                 * 00880000
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***       * 00890000
//********************************************************************* 00900000
//COMINIT EXEC COMINIT,                                                 00910000
//             LOADLIB='INFORM.LOADLIB',                                00920000
//             INFCOM='INFORM.INFCOM'                                   00930000
//DEFCOM.SYSIN DD *                                                     00940000
      DELETE ('INFORM.INFCOM') CLUSTER PURGE                            00950000
      DEFINE CLUSTER -                                                  00960000
               (NAME('INFORM.INFCOM') -                                 00970000
               VOL('VOLUME') -                                          00980000
               RECORDS(409) -                                           00990000
               SHAREOPTIONS(3 3) -                                      01000000
               WRITECHECK NUMBERED -                                    01010000
               OWNER('USER')) -                                         01020000
             DATA(NAME('INFORM.INFCOM.DATA') -                          01030000
               CONTROLINTERVALSIZE(4096) -                              01040000
               RECORDSIZE(4088 4088) -                                  01050000
               BUFFERSPACE(8192))                                       01060000
//INITCOM.INFIN DD *                                                    01070000
ULSCOM                                                                  01080000
//********************************************************************* 01090000
//*  FOR BGINIT, ALSO CHANGE THE DEFBG.SYSIN STATEMENTS:              * 01100000
//*                                                                   * 01110000
//*      NAME('INFORM.BGLIB')                                         * 01120000
```

Figure 10-3 Sample Execution JCL for the Initialization Utility, INIT2 (Page 2 of 3)

```
//*      NAME('INFORM.BGLIB.DATA')                                     * 01130000
//*      VOL('VOLUME')                                                 * 01140000
//*      OWNER('USER')                                                 * 01150000
//*                                                                    * 01160000
//*  REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR INFORMATION    * 01170000
//*  ABOUT CHANGING THE DCB ATTRIBUTES.                                * 01180000
//*                                                                    * 01190000
//*  *** DO NOT MAKE ANY CHANGES IN INITBG.M4INPUT ***                 * 01200000
//*  *** DELETE CLUSTER STMTS ARE PROVIDED FOR RERUNS ONLY. ***        * 01210000
//********************************************************************** 01220000
//BGINIT  EXEC BGINIT,                                                   01230000
//             LOADLIB='INFORM.LOADLIB',                                 01240000
//             BGLIB='INFORM.BGLIB'                                      01250000
//DEFBG.SYSIN DD *                                                       01260000
      DELETE  'INFORM.BGLIB' CLUSTER PURGE                               01270000
      DEFINE CLUSTER -                                                   01280000
               (NAME('INFORM.BGLIB') -                                   01290000
               VOL('VOLUME') -                                           01300000
               CYLINDERS(1 1) -                                          01310000
               SHAREOPTIONS(3 3) -                                       01320000
               CONTROLINTERVALSIZE(4096) -                               01330000
               RECORDSIZE(507 507) -                                     01340000
               WRITECHECK NUMBERED -                                     01350000
               OWNER('USER')) -                                          01360000
             DATA(NAME('INFORM.BGLIB.DATA'))                             01370000
//INITBG.M4INPUT DD *                                                    01380000
        UCINIT                                                           01390000
```

Figure 10-3 Sample Execution JCL for the Initialization Utility, INIT2 (Page 3 of 3)

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the Initialization Utility JCL procedure, INIT or INIT2.

Variable name	Description
INFCOM	Specify the name of the communication file.
LOADLIB	Specify the VISION:Inform installation load library to be used by the Initialization Utility.
BGLIB	Specify the name of the background library.
FGLIB	Specify the name of the foreground library.

Figure 10-4 JCL Procedure Variables for the INIT Utility

Providing DD Statement Overrides

Provide the following DD statement overrides when executing the supplied Initialization Utility.

Note: In the following examples, you modify text in lowercase letters. Do not change the text in uppercase letters.

FGLIB.SYSIN for Foreground Library Delete and Define

The following figure shows FGLIB.SYSIN sample input.

```
DELETE 'inform.fglib' CLUSTER PURGE
      DEFINE CLUSTER -
               (NAME('inform.fglib') -
               VOL('volume') -
               RECORDS(nnn) -
               SHAREOPTIONS(2 3) -
               WRITECHECK NUMBERED -
               OWNER('user')) -
             DATA(NAME('inform.fglib.DATA') -
               CONTROLINTERVALSIZE(4096) -
               RECORDSIZE(mmm mmm))
```

Figure 10-5 FGLIB.SYSIN Sample Input for Foreground Library Delete and Define

Note: The example shown in figure 10-5 is from the INIT JCL. INIT2 is slightly different, with different VSAM SHAREOPTION and BUFFERSPACE added. The items to be changed (lowercase entries) are the same for both.

Provide the IDCAMS statements to delete the old foreground library and define the new foreground library in FGLIB.SYSIN.

1. Replace the two occurrences of 'inform.fglib' with the name of the foreground library.
2. Replace ‘volume’ with the volume serial number of the disk device for the foreground library.
3. Replace 'nnn' with the expected number of blocks.The default is 385.

   If you need a different number of blocks, change the QFILE parameter in the PARMBLK.
4. Replace ‘user’ with any valid characters as specified for the IDCAMS parameter definitions.
5. Replace 'mmm mmm' with the expected block size or control interval size. The default is 4088. If you need a different block size, change the QFILE parameter in the PARMBLK.
6. In Job INIT2 only, make BUFFERSIZE exactly twice the size of the control interval.

Note: For more information on how to calculate the number of blocks (records) and code PARMBLK parameters, refer to the *Advantage VISION:Inform Installation Guide*.

DEFCOM.SYSIN for Communication File Delete and Define

The following figure shows DEFCOM.SYSIN sample input.

```
DELETE 'inform.infcom' CLUSTER PURGE
      DEFINE CLUSTER -
                (NAME('inform.infcom') -
                VOL('volume') -
                RECORDS(nnn) -
                SHAREOPTIONS(3 3) -
                WRITECHECK NUMBERED -
                OWNER('user')) -
             DATA(NAME('inform.infcom.DATA') -
                CONTROLINTERVALSIZE(4096) -
                RECORDSIZE(mmm mmm) -
                BUFFERSPACE (8192))
```

Figure 10-6 DEFCOM.SYSIN Sample Input for the Communication File Delete and Define

Provide the IDCAMS statements to delete the old communication file and define the new communication file in DEFCOM.SYSIN.

1. Replace the two occurrence of 'inform.infcom' with the name of the communication file.
2. Replace ‘volume’ with the volume serial number of the disk device for the communication file.
3. Replace 'nnn' with the expected number of blocks.The default is 409.

   If you need a different number of blocks, change the QFILE parameter in the PARMBLK.
4. Replace ‘user’ with any valid character as defined for the IDCAMS parameter definitions.
5. Replace 'mmm mmm' with the expected block size or control interval size. The default is 4088. If you need a different block size, change the QFILE parameter in the PARMBLK.
6. Ensure that BUFFERSIZE is exactly twice the size of the control interval.

Note: For more information on how to calculate the number of blocks (records) and code PARMBLK parameters, see the *Advantage VISION:Inform Installation Guide*.

DEFBG.SYSIN for Background Library Delete and Define

The following figure shows the DEFBG.SYSIN sample input.

```
DELETE 'inform.bglib' CLUSTER PURGE
DEFINE CLUSTER -
                (NAME('inform.bglib') -
                VOL('volume') -
                CYL(1 1) -
                SHAREOPTIONS(3 3) -
                CONTROLINTERVALSIZE(4096) -
                NUMBERED WRITECHECK -
                RECORDSIZE(nnn nnn) -
                OWNER('user')) -
            DATA(NAME('inform.bglib.DATA'))
```

Figure 10-7 DEFBG.SYSIN Sample Input for the Background Delete and Define

Provide the IDCAMS statements to delete the old background library and define the new background library in DEFBG.SYSIN.

1. Replace the two occurrences of 'inform.bglib' with the name of the background library.
2. Replace ‘volume’ with the volume serial number for the disk device for the background library.
3. Replace 'nnn nnn' with the expected record size. The minimum is 507.
4. Replace ‘user’ with any valid characters as specified for IDCAMS parameter definitions.
5. If you need a different record size, use the following guidelines:

Directory Blocking Factor

The directory blocking factor (record size or number of directory detail records in a DDR) is determined for VSAM common libraries by the record size specified in the cluster definition.

- The record size must be at least 507 bytes, but could be larger. A 507-byte record size means that all records on the common library are 507 bytes.
- A 507-byte DDR allows 15 directory entries (each entry is 32 bytes) of which 12 point to data records or items.
- A larger record size increases the directory blocking factor. Each additional 32 bytes increases the directory entries by 1.
- The greater the blocking factor, the fewer accesses there are to locate an item on the common library.

Best Record Size

The best record size is the one that is closest to the average size of a common library item and which fits without excessive waste in the chosen control interval size (CI size).

- Note the allowable directory blocking factor for the record size.
- Choose a CI size that best balances a block size which is adequately large and data transfer time that does not stall the processing throughput. CI sizes of 2048 (2K) and 4096 (4K) are good CI sizes.
- VSAM is likely to compute a CI size of 2048 if not overridden in the cluster.

With a record size of 507 and CI size of 2048, four records are stored in the CI.

- Each CI contains VSAM control information; a 4-byte CIDF (one per CI) and a 3-byte RDF (one per record).
- Thus, four records in the CI are 2028 bytes (4 times 507), plus 16 bytes of VSAM control fields (4 for the CIDF and 12 for the RDFs).
- So with the 2048 CI size and 507 record size, 2044 bytes of the CI are used and 4 bytes unused.

You can best decide the CI size and record size after discovering the average size of your common library items, which will vary from one installation to another. Determine the average from a dump of an existing common library.

- Directory records are easily spotted and bytes 1 through 4 of each directory detail entry contain the binary length of the data item.
- For a VSAM common library, the length of a data item can exceed both record and CI length.

Optimally, however, the record size and longest data item should not exceed the CI size.

Generated Output

The Initialization Utility produces the following informational listings:

FGLIB.INFPRINT	This listing summarizes the results of the foreground library initialization process and contains the library's space allocation.
DEFCOM.INFPRINT	This listing summarizes the results of the communication file initialization process and contains the file's space allocation.
INITBG.M4LIST	This listing summarizes the results of the background library initialization process and contains the library's space allocation.

Chapter 11

Library Backup Utility (LBBACKUP)

The Library Backup Utility, LBBACKUP, creates backup copies of the background and foreground libraries.

The resulting sequential data sets contain the file definitions, table definitions, field descriptions, alternate names, and user profiles. The utility also copies QUERY and STMTS items under the user profiles.

Computer Associates recommends that you back up your VISION:Inform libraries regularly.

Utility Component

The Library Backup Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(LBBACKUP)

This sample JCL to execute the Library Backup Utility is in the VISION:Inform JCL library, member LBBACKUP.

Flow Diagram

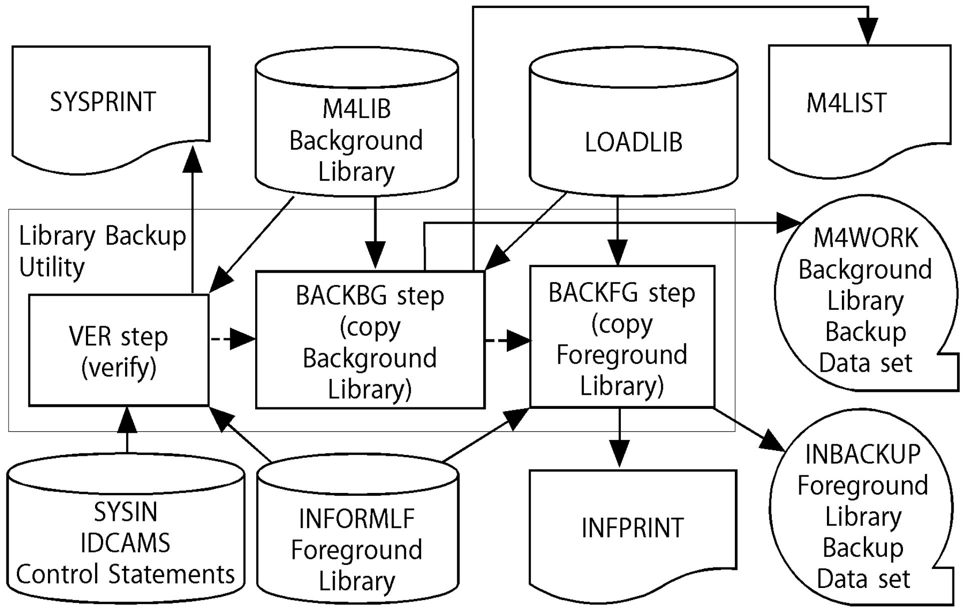
SYSPRINT
M4LIB Background Library
LOADLIB
M4LIST
Library Backup Utility
VER step (verify)
BACKBG step (copy Background Library)
BACKFG step (copy Foreground Library)
M4WORK Background Library Backup Data set
SYSIN IDCAMS Control Statements
INFORMLF Foreground Library
INFPRINT
INBACKUP Foreground Library Backup Data set

Figure 11-1 Library Backup Utility Flow Diagram

Utility Flow

The Library Backup Utility, LBBACKUP, consists of the following steps:

1. VER step

   The verify step executes the VSAM IDCAMS utility to verify that all the required VSAM files are available to the backup job. The following files are verified:

   – VISION:Inform foreground library.

   – VISION:Inform background library.

   The utility creates the SYSOUT data set, VER.SYSPRINT.

   The acceptable return code for this step is 0.

2. BACKBG step

   This step creates a sequential copy of the background library.

   The utility creates the SYSOUT data set, BACKBG.M4LIST.

   The acceptable return code for this step is 0.

3. BACKFG step

   This step creates a sequential copy of the foreground library.

   The utility creates the SYSOUT data set, BACKFG.INFPRINT.

   The acceptable return code for this step is 0.

Note: Utility flow step names correspond to JCL job step names.

Executing the Utility

The Library Backup Utility, LBBACKUP, is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(LBBACKUP)

This JCL contains an instream procedure, followed by the steps to execute this procedure.

Specifying the Execution JCL

Figure 11-2 shows the sample JCL contained in INFORM.JCL in the member LBBACKUP. Review this JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the Library Backup Utility.

```
//* MEMBER LBBACKUP
//*********************************************************************
//*   THIS PROCEDURE CREATES BACKUP COPIES OF THE BACKGROUND LIBRARY  *
//*   AND FOREGROUND LIBRARY ON A TAPE DEVICE.                        *
//*********************************************************************
//LBBACKUP PROC FGLIB=,
//            BGLIB=,
//            LOADLIB=,
//            NAMEFG=,
//            FGFILNO=,
//            NAMEBG=,
//            BGFILNO=,
//            UNIT=,
//            VOLSER=
//VER    EXEC PGM=IDCAMS,REGION=512K
//FGLIB    DD DISP=SHR,DSN=&FGLIB
//BGLIB    DD DISP=SHR,DSN=&BGLIB
//SYSPRINT DD SYSOUT=*
//BACKBG EXEC PGM=MARKUTIL,REGION=512K
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB
//M4LIST   DD SYSOUT=*
//M4LIB    DD DISP=SHR,DSN=&BGLIB
//M4WORK   DD DISP=(,CATLG,DELETE),DSN=&NAMEBG,
//            UNIT=&UNIT,VOL=SER=&VOLSER,LABEL=(&BGFILNO,SL)
//BACKFG EXEC PGM=INFORMUD,REGION=512K
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB
//INFPRINT DD SYSOUT=*
//INFORMLF DD DISP=SHR,DSN=&FGLIB
//INBACKUP DD DISP=(NEW,KEEP),DSN=&NAMEFG,UNIT=&UNIT,VOL=SER=&VOLSER,
//            LABEL=(&FGFILNO,SL)
//       PEND
//********************************************************************
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           *
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          *
//*                                                                  *
```

Figure 11-2 Sample Execution JCL for the Library Backup Utility, LBBACKUP (Page 1 of 2)

```
//* FGLIB    - THE INFORM FOREGROUND LIBRARY.                       *
//* BGLIB    - THE INFORM BACKGROUND LIBRARY.                       *
//* LOADLIB  - THE INFORM INSTALLATION LOAD LIBRARY.                *
//* NAMEFG   - NAME OF THE FOREGROUND LIBRARY BACKUP DATA SET.      *
//* FGFILNO  - THE NUMBER OF THE TAPE FILE TO CONTAIN THE BACKUP OF  *
//*            THE FOREGROUND LIBRARY.  THE DEFAULT IS 2.           *
//* NAMEBG   - NAME OF THE BACKGROUND LIBRARY BACKUP DATA SET.      *
//* BGFILNO  - THE NUMBER OF THE TAPE FILE TO CONTAIN THE BACKUP OF  *
//*            THE BACKGROUND LIBRARY.  THE DEFAULT IS 1.           *
//* UNIT     - THE UNIT OF THE BACKUP FILE. THE DEFAULT IS TAPE.    *
//* VOLSER   - THE VOLUME SERIAL NUMBER OF THE BACKUP FILE.         *
//*                                                                 *
//* *** NOTE *** DO NOT ALTER THE BACKBG.M4INPUT STATEMENT.         *
//*******************************************************************
//BACKUP  EXEC LBBACKUP,
//             FGLIB='INFORM.FGLIB',
//             BGLIB='INFORM.BGLIB',
//             LOADLIB='INFORM.LOADLIB',
//             NAMEFG='INFORM.BACKUP.FGLIB',
//             FGFILNO=2,
//             NAMEBG='INFORM.BACKUP.BGLIB',
//             BGFILNO=1,
//             UNIT=TAPE,
//             VOLSER=XXXXXX
//VER.SYSIN DD *
         VERIFY FILE(FGLIB)
         VERIFY FILE(BGLIB)
//BACKBG.M4INPUT DD *
        UCDUMP
```

Figure 11-2 Sample Execution JCL for the Library Backup Utility, LBBACKUP (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the Library Backup Utility JCL procedure, LBBACKUP.

Variable name	Description
FGLIB	Specify the name of the foreground library to be backed up.
BGLIB	Specify the name of the background library to be backed up.
LOADLIB	Specify the name of the VISION:Inform installation load library to be used by the utility.
NAMEFG	Specify the name of the foreground library backup data set.
FGFILNO	Specify the number of the tape file to contain the backup of the foreground library. The default is 2.
NAMEBG	Specify the name of the background library backup data set.
BGFILNO	Specify the number of the tape file to contain the backup of the background library. The default is 1.
UNIT	Specify the unit of the backup file. The default is tape.
VOLSER	Specify the volume serial number of the backup file.

Figure 11-3 JCL Procedure Variables for the LBBACKUP Utility

Generated Output

The Library Backup Utility generates the following informational listings:

BACKBG.M4LIST	This listing summarizes the results of the background library backup process. It contains a list of the members that were copied to the backup data set.
BACKFG.INFPRINT	This listing summarizes the results of the foreground library backup process. It contains a list of the members that were copied to the backup data set.

Invalid Internal Pointer

The following message in the INFPRINT listing means that a member (file definition, user profile, and so on) has an invalid internal pointer.

JX01** Member ******** truncated due to bad pointer **.

If the Library Backup Utility issues this message, run the Library Restore Utility to fix the bad pointer.

Note that the Library Restore Utility fixes broken pointers, but some data might be lost.

Space Utilization Statistics

Each time you run the VISION:Inform Library Backup Utility against the foreground library, the utility generates space utilization statistics and writes data to the INFPRINT summary listing.

A sample of the actual message text follows. The asterisks in the message text indicate where the utility inserts numerical values for each run.

```
**S801 Of ******* overflow blocks, ******* are free, and ******* are in use.
**S802 Of ******* root blocks, ******* are free, and ******* are in use.
**S803 Of ******* root blocks, ******* have overflowed.
```

With these space utilization statistics, the system administrator can now determine:

- When to allocate additional space for the library (usually when the number of available overflow blocks becomes too small).
- When to increase the number of root blocks (usually when the number of available root blocks becomes too small).
- When to increase the block sizes (usually when the number of overflow blocks becomes too high).
- When to increase the bucket sizes (usually when the number of overflow blocks becomes too high).

Chapter 12

Library Restore Utility (LBRESTOR)

The Library Restore Utility, LBRESTOR, restores and reorganizes the foreground and background libraries up to the last backup date. The Library Restore Utility deletes an old library, defines and initializes a new library, then restores the contents from the backup data set. Items that were added after the last backup will be lost and will have to be recreated.

Note: Take the foreground library offline to run this utility.

The input to the Library Restore Utility are the sequential data sets created by the Library Backup Utility, LBBACKUP.

Utility Component

The Library Restore Utility is composed of the following components:

Sample Execution JCL — INFORM.JCL(LBRESTOR)

This sample JCL, which executes the Library Restore Utility is in the VISION:Inform JCL library, member LBRESTOR. This version restores a foreground library with VSAM SHAREOPTIONS (2,3). This member corresponds to the INIT version of the Initialization Utility.

Sample Execution JCL — INFORM.JCL(LBREST2)

This sample JCL, which executes the Library Restore Utility, is in the VISION:Inform JCL library, member, LBREST2. This version restores a foreground library with VSAM SHAREOPTIONS (3,3). This member corresponds to the INIT2 version of the Initialization Utility.

Flow Diagram

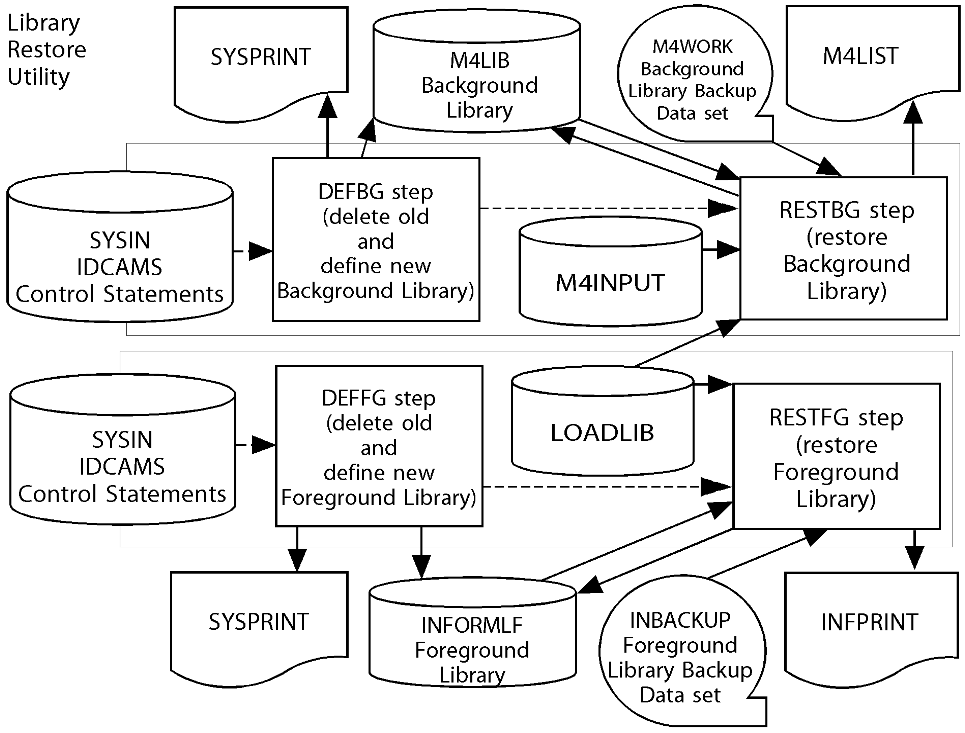
Library Restore Utility
SYSPRINT
M4LIB Background Library
M4WORK Background Library Backup Data set
M4LIST
SYSIN IDCAMS Control Statements
DEFBG step (delete old and define new Background Library)
M4INPUT
RESTBG step (restore Background Library)
SYSIN IDCAMS Control Statements
DEFFG step (delete old and define new Foreground Library)
LOADLIB
RESTFG step (restore Foreground Library)
SYSPRINT
INFORMLF Foreground Library
INBACKUP Foreground Library Backup Data set
INFPRINT

Figure 12-1 Library Restore Utility Flow Diagram

Utility Flow

The Library Restore Utility, LBRESTOR, or LBREST2, consists of the following steps:

1. DEFBG step

   This step executes the VSAM IDCAMS utility to delete the old background library and define the new background library.

   The utility creates the SYSOUT data set, DEFBG.SYSPRINT.

   The acceptable return code for this step is 0.
2. RESTBG step

   This step restores the background library.

   The utility creates the SYSOUT data set, RESTBG.M4LIST.

   The acceptable return code for this step is 0.
3. DEFFG step

   This step executes the VSAM IDCAMS utility to delete the old foreground library and define the new foreground library.

   The utility creates the SYSOUT data set, DEFFG.SYSPRINT.

   The acceptable return code for this step is 0.
4. RESTFG step

   This step restores the foreground library.

   The utility creates the SYSOUT data set, RESTFG.INFPRINT.

   The acceptable return code for this step is 0.

Note: Utility flow step names correspond to JCL job step names.

For more information about using LBRESTOR or LBREST2, see the *Advantage VISION:Inform for CICS Installation Guide.*

Executing the Utility

The Library Restore Utility, LBRESTOR, is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(LBRESTOR) or INFORM.JCL(LBREST2)

Note: Take the foreground library offline to run this utility.

This JCL contains an instream procedure, followed by the steps for this procedure.

Specifying the Execution JCL

Figure 12-2 shows the sample JCL contained in INFORM.JCL in the member LBRESTOR. Review this JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply procedure variable values—for information, see Specifying Procedure Variables.
3. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the Library Restore Utility. Use this member if you use the INIT version of the Initialization Utility.

```
//* MEMBER LBRESTOR                                                     00010000
//********************************************************************* 00020000
//* THIS PROCEDURE RESTORES THE FOREGROUND AND BACKGROUND LIBRARIES   * 00030000
//* FROM BACKUP DATASETS CREATED BY THE LBBACKUP JOB.  THIS JOB       * 00040000
//* ASSUMES THAT THE BACKUP FILES ARE ON TAPE.                        * 00050000
//********************************************************************* 00060000
//LBRESTOR PROC FGLIB=,                                                 00070000
//           BGLIB=,                                                    00080001
//           LOADLIB=,                                                  00090001
//           NAMEBG=,                                                   00100001
//           NAMEFG=,                                                   00110001
//           BGFILNO=,                                                  00120001
//           FGFILNO=,                                                  00130001
//           UNIT=,                                                     00140001
//           VOLSER=                                                    00150001
//DEFBG  EXEC PGM=IDCAMS,REGION=512K                                    00160001
//SYSPRINT DD  SYSOUT=*                                                 00170000
//RESTBG EXEC PGM=MARKUTIL,REGION=512K                                  00180001
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00190003
//M4LIB    DD DISP=OLD,DSN=&BGLIB                                       00200003
//M4LIST   DD SYSOUT=*                                                  00210001
//M4WORK   DD DISP=OLD,DSN=&NAMEBG,LABEL=(&BGFILNO,SL),                 00220003
//           UNIT=&UNIT,VOL=SER=&VOLSER                                 00230001
//DEFFG  EXEC PGM=IDCAMS,REGION=512K                                    00240001
//SYSPRINT DD SYSOUT=*                                                  00250001
//RESTFG EXEC PGM=INFORMUR,REGION=512K                                  00260001
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00270001
//INFPRINT DD SYSOUT=*                                                  00280001
//INFORMLF DD DISP=SHR,DSN=&FGLIB                                       00290001
//INBACKUP DD DISP=OLD,DSN=&NAMEFG,UNIT=&UNIT,VOL=SER=&VOLSER,          00300001
//           LABEL=(&FGFILNO,SL)                                        00310001
//       PEND                                                           00320001
//********************************************************************* 00330000
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.            * 00340002
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                           * 00350001
//*                                                                   * 00360000
//* FGLIB    - THE INFORM FOREGROUND LIBRARY.                         * 00370001
//* BGLIB    - THE INFORM BACKGROUND LIBRARY.                         * 00380001
//* LOADLIB  - THE INFORM INSTALLATION LOAD LIBRARY.                  * 00390001
//* NAMEBG   - NAME OF THE BACKUP FILE FOR THE BACKGROUND LIBRARY.    * 00400000
//* NAMEFG   - NAME OF THE BACKUP FILE FOR THE FOREGROUND LIBRARY.    * 00410001
//* BGFILNO  - THE NUMBER OF THE TAPE FILE CONTAINING THE BACKGROUND  * 00420000
//*            LIBRARY BACKUP DATA.  DEFAULT = 1.                     * 00430000
//* FGFILNO  - THE NUMBER OF THE TAPE FILE CONTAINING THE FOREGROUND  * 00440000
//*            LIBRARY BACKUP DATA.  DEFAULT = 2.                     * 00450000
//* UNIT     - THE UNIT OF THE BACKUP FILE. THE DEFAULT IS TAPE.      * 00460000
//* VOLSER   - THE VOLUME SERIAL NUMBER OF THE BACKUP FILE.           * 00470000
//*                                                                   * 00480000
//*   CHANGE THE DEFFG.SYSIN SPECIFICATION AS YOU DID FOR THE         * 00490000
//*   INIT PROCEDURE.                                                 * 00500000
//*                   NAME('INFORM.FGLIB')                            * 00510001
//*                   NAME('INFORM.FGLIB.DATA')                       * 00520001
//*                   VOL('VOLUME')                                   * 00530000
```

Figure 12-2 Sample Execution JCL for the Library Restore Utility, LBRESTOR (Page 1 of 2)

```
//*                   OWNER('USER')                                   * 00540000
//*                                                                   * 00550000
//* CHANGE THE DEFBG.SYSIN SPECIFICATIONS:                            * 00560000
//*                                                                   * 00570000
//*                   NAME('INFORM.BGLIB')                            * 00580001
//*                   VOL('VOLUME')                                   * 00590000
//*                   OWNER('USER')                                   * 00600000
//*                   NAME('INFORM.BGLIB.DATA')                       * 00610001
//*                                                                   * 00620000
//*   REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR MORE         * 00630000
//*   INFORMATION.                                                    * 00640000
//********************************************************************* 00650000
//* **** NOTE **** DO NOT ALTER THE RESTBG.M4INPUT STATEMENT.         * 00660000
//********************************************************************* 00670000
//RESTORE EXEC LBRESTOR,                                                00680000
//             FGLIB='INFORM.FGLIB',                                    00690001
//             BGLIB='INFORM.BGLIB',                                    00700001
//             LOADLIB='INFORM.LOADLIB',                                00710001
//             NAMEBG='INFORM.BACKUP.BGLIB',                            00720001
//             NAMEFG='INFORM.BACKUP.FGLIB',                            00730001
//             BGFILNO=1,                                               00740000
//             FGFILNO=2,                                               00750000
//             UNIT=TAPE,                                               00760000
//             VOLSER=XXXXXX                                            00770000
//DEFBG.SYSIN DD *                                                      00780000
       DELETE 'INFORM.BGLIB' CLUSTER PURGE                              00790001
       DEFINE CLUSTER -                                                 00800000
               (NAME('INFORM.BGLIB') -                                  00810001
               VOL('VOLUME') -                                          00820000
               CYL(1 1) -                                               00830000
               SHAREOPTIONS(3 3) -                                      00840000
               CONTROLINTERVALSIZE(4096) -                              00850000
               NUMBERED WRITECHECK -                                    00860000
               RECORDSIZE(507 507) -                                    00870000
               OWNER('USER')) -                                         00880000
             DATA(NAME('INFORM.BGLIB.DATA'))                            00890001
//RESTBG.M4INPUT DD *                                                   00900000
        UCREST                                                          00910000
//DEFFG.SYSIN DD *                                                      00920000
      DELETE 'INFORM.FGLIB' CLUSTER PURGE                               00930001
      DEFINE CLUSTER -                                                  00940000
               (NAME('INFORM.FGLIB') -                                  00950001
               VOL('VOLUME') -                                          00960000
               RECORDS(385) -                                           00970000
               SHAREOPTIONS(2 3) -                                      00980000
               WRITECHECK NUMBERED -                                    00990000
               OWNER('USER')) -                                         01000000
             DATA(NAME('INFORM.FGLIB.DATA') -                           01010001
               CONTROLINTERVALSIZE(4096) -                              01020000
               RECORDSIZE(4088 4088))                                   01030000
```

Figure 12-2 Sample Execution JCL for the Library Restore Utility, LBRESTOR (Page 2 of 2)

The following figure shows the member LBREST2 from the INFORM.JCL library. Use this member if you used the INIT2 version of the Initialization Utility. This version of the Library Restore Utility restores the foreground library with VSAM SHAREOPTIONS of (3,3).

```
//* MEMBER LBREST2                                                      00010000
//********************************************************************* 00020000
//* THIS PROCEDURE RESTORES THE FOREGROUND AND BACKGROUND LIBRARIES   * 00030000
//* FROM BACKUP DATASETS CREATED BY THE LBBACKUP JOB.  THIS JOB       * 00040000
//* ASSUMES THAT THE BACKUP FILES ARE ON TAPE.                        * 00050000
//* THIS JOB SHOULD BE USED IF YOU ARE USING THE HOSTCONNECT SERVER   * 00060000
//* CLIENT OR IF YOU DESIRE TO ESTABLISH CONCURRENT UPDATE ACCESS TO  * 00070000
//* THE FOREGROUND LIBRARY.                                           * 00080000
//********************************************************************* 00090000
//LBRESTOR PROC FGLIB=,                                                 00100000
//            BGLIB=,                                                   00110000
//            LOADLIB=,                                                 00120000
//            NAMEBG=,                                                  00130000
//            NAMEFG=,                                                  00140000
//            BGFILNO=,                                                 00150000
//            FGFILNO=,                                                 00160000
//            UNIT=,                                                    00170000
//            VOLSER=                                                   00180000
//DEFBG  EXEC PGM=IDCAMS,REGION=512K                                    00190000
//SYSPRINT DD  SYSOUT=*                                                 00200000
//RESTBG EXEC PGM=MARKUTIL,REGION=512K                                  00210000
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00220000
//M4LIB    DD DISP=OLD,DSN=&BGLIB                                       00230000
//M4LIST   DD SYSOUT=*                                                  00240000
//M4WORK   DD DISP=OLD,DSN=&NAMEBG,LABEL=(&BGFILNO,SL),                 00250000
//            UNIT=&UNIT,VOL=SER=&VOLSER                                00260000
//DEFFG  EXEC PGM=IDCAMS,REGION=512K                                    00270000
//SYSPRINT DD SYSOUT=*                                                  00280000
//RESTFG EXEC PGM=INFORMUR,REGION=512K                                  00290000
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00300000
//INFPRINT DD SYSOUT=*                                                  00310000
//INFORMLF DD DISP=SHR,DSN=&FGLIB                                       00320000
//INBACKUP DD DISP=OLD,DSN=&NAMEFG,UNIT=&UNIT,VOL=SER=&VOLSER,          00330000
//            LABEL=(&FGFILNO,SL)                                       00340000
//       PEND                                                           00350000
//********************************************************************  00360000
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           *  00370000
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          *  00380000
//*                                                                  *  00390000
//* FGLIB    - THE INFORM FOREGROUND LIBRARY.                        *  00400000
//* BGLIB    - THE INFORM BACKGROUND LIBRARY.                        *  00410000
//* LOADLIB  - THE INFORM INSTALLATION LOAD LIBRARY.                 *  00420000
//* NAMEBG   - NAME OF THE BACKUP FILE FOR THE BACKGROUND LIBRARY.   *  00430000
//* NAMEFG   - NAME OF THE BACKUP FILE FOR THE FOREGROUND LIBRARY.   *  00440000
//* BGFILNO  - THE NUMBER OF THE TAPE FILE CONTAINING THE BACKGROUND *  00450000
//*            LIBRARY BACKUP DATA.  DEFAULT = 1.                    *  00460000
//* FGFILNO  - THE NUMBER OF THE TAPE FILE CONTAINING THE FOREGROUND *  00470000
//*            LIBRARY BACKUP DATA.  DEFAULT = 2.                    *  00480000
//* UNIT     - THE UNIT OF THE BACKUP FILE. THE DEFAULT IS TAPE.     *  00490000
//* VOLSER   - THE VOLUME SERIAL NUMBER OF THE BACKUP FILE.          *  00500000
//*                                                                  *  00510000
//*   CHANGE THE DEFFG.SYSIN SPECIFICATION AS YOU DID FOR THE        *  00520000
//*   INIT PROCEDURE.                                                *  00530000
//*                   NAME('INFORM.FGLIB')                           *  00540000
//*                   NAME('INFORM.FGLIB.DATA')                      *  00550000
//*                   VOL('VOLUME')                                  *  00560000
//*                   OWNER('USER')                                  *  00570000
//*                                                                  *  00580000
//* CHANGE THE DEFBG.SYSIN SPECIFICATIONS:                           *  00590000
//*                                                                  *  00600000
//*                   NAME('INFORM.BGLIB')                           *  00610000
//*                   VOL('VOLUME')                                  *  00620000
```

Figure 12-3   Sample Execution JCL for the Library Restore Utility, LBREST2 (Page 1 of 2)

```
//*                   OWNER('USER')                                  *  00630000
//*                   NAME('INFORM.BGLIB.DATA')                      *  00640000
//*                                                                  *  00650000
//*   REFER TO THE IBM ACCESS METHOD SERVICES MANUAL FOR MORE        *  00660000
//*   INFORMATION.                                                   *  00670000
//********************************************************************  00680000
//* **** NOTE **** DO NOT ALTER THE RESTBG.M4INPUT STATEMENT.        *  00690000
//********************************************************************  00700000
//RESTORE EXEC LBRESTOR,                                                00710000
//             FGLIB='INFORM.FGLIB',                                    00720000
//             BGLIB='INFORM.BGLIB',                                    00730000
//             LOADLIB='INFORM.LOADLIB',                                00740000
//             NAMEBG='INFORM.BACKUP.BGLIB',                            00750000
//             NAMEFG='INFORM.BACKUP.FGLIB',                            00760000
//             BGFILNO=1,                                               00770000
//             FGFILNO=2,                                               00780000
//             UNIT=TAPE,                                               00790000
//             VOLSER=XXXXXX                                            00800000
//DEFBG.SYSIN DD *                                                      00810000
       DELETE 'INFORM.BGLIB' CLUSTER PURGE                              00820000
       DEFINE CLUSTER -                                                 00830000
               (NAME('INFORM.BGLIB') -                                  00840000
               VOL('VOLUME') -                                          00850000
               CYL(1 1) -                                               00860000
               SHAREOPTIONS(3 3) -                                      00870000
               CONTROLINTERVALSIZE(4096) -                              00880000
               NUMBERED WRITECHECK -                                    00890000
               RECORDSIZE(507 507) -                                    00900000
               OWNER('USER')) -                                         00910000
             DATA(NAME('INFORM.BGLIB.DATA'))                            00920000
//RESTBG.M4INPUT DD *                                                   00930000
        UCREST                                                          00940000
//DEFFG.SYSIN DD *                                                      00950000
      DELETE 'INFORM.FGLIB' CLUSTER PURGE                               00960000
      DEFINE CLUSTER -                                                  00970000
               (NAME('INFORM.FGLIB') -                                  00980000
               VOL('VOLUME') -                                          00990000
               RECORDS(385) -                                           01000000
               SHAREOPTIONS(3 3) -                                      01020000
               WRITECHECK NUMBERED -                                    01030000
               OWNER('USER')) -                                         01040000
             DATA(NAME('INFORM.FGLIB.DATA') -                           01050000
               CONTROLINTERVALSIZE(4096) -                              01060000
               RECORDSIZE(4088 4088) -                                  01070001
               BUFFERSPACE(8192))                                       01080001
```

Figure 12-3 Sample Execution JCL for the Library Restore Utility, LBREST2 (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the Library Restore Utility JCL procedure, LBRESTOR or LBREST2.

Variable name	Description
FGLIB	Specify the name of the foreground library to be restored.
BGLIB	Specify the name of the background library to be restored.
LOADLIB	Specify the name of the VISION:Inform installation load library to be used by the utility.
NAMEFG	Specify the name of the foreground library backup data set.
FGFILNO	Specify the number of the tape file containing the foreground library backup data. The default is 2.
NAMEBG	Specify the name of the background library backup data set.
BGFILNO	Specify the number of the tape file containing background library backup data. The default is 1.
UNIT	Specify the unit of the backup file. The default is tape.
VOLSER	Specify the volume serial number of the backup file.

Figure 12-4 JCL Procedure Variables for the LBRESTOR Utility

Providing DD Statement Overrides

Provide the following DD statement overrides when executing the Library Restore Utility.
Note: In the following examples, you modify text in lowercase letters.
Do not change the text in uppercase letters.

DEFBG.SYSIN for Background Library Delete and Define

The following figure shows the DEFBG.SYSIN sample input.

```
DELETE 'inform.bglib' CLUSTER PURGE
        DEFINE CLUSTER -
               (NAME('inform.bglib') -
               VOL('volume') -
               CYL(1 1) -
               SHAREOPTIONS(3 3) -
               CONTROLINTERVALSIZE(4096) -
               NUMBERED WRITECHECK -
               RECORDSIZE(nnn nnn) -
                OWNER('user')) -
              DATA(NAME('inform.bglib.DATA'))
```

Figure 12-5 DEFBG.SYSIN Sample Input for the Background Library Delete and Define

Provide the IDCAMS control statements to delete the old background library and define the new background library in DEFBG.SYSIN.

1. Replace the two occurrences of 'inform.bglib' with the name of the background library.
2. Replace ‘volume’ with the volume serial number of the disk device for the background library.
3. Replace 'nnn nnn' with the expected record size. The minimum is 507.

   For a detailed explanation on how to calculate the record size, see Chapter 10, *Initialization Utility (INIT)*.
4. Replace ‘user’ with any valid characters as specified for the IDCAMS parameter definitions.

DEFFG.SYSIN for Foreground Library Delete and Define

The following figure shows DEFFG.SYSIN sample input.

```
DELETE 'inform.fglib' CLUSTER PURGE
      DEFINE CLUSTER -
               (NAME('inform.fglib') -
               VOL('volume') -
               RECORDS(nnn) -
               SHAREOPTIONS(2 3) -
               WRITECHECK NUMBERED -
               OWNER('user')) -
             DATA(NAME('inform.fglib.DATA') -
               CONTROLINTERVALSIZE(4096) -
               RECORDSIZE(mmm mmm))
```

Figure 12-6 DEFFG.SYSIN Sample Input for the Foreground Library Delete and Define

Note: This sample is from LBRESTOR. The alternate version, LBREST2, has a different SHAREOPTIONS value and an additional BUFFERSPACE parameter. The required changes, in lowercase, are the same for both procedures.

Provide the IDCAMS control statements to delete the old foreground library and define the new foreground library in DEFFG.SYSIN.

1. Replace the two occurrences of 'inform.fglib' with the name of the foreground library.
2. Replace ‘volume’ with the volume serial number of the disk device for the foreground library.
3. Replace 'nnn' with the expected number of blocks.The default is 385.
4. Replace ‘user’ with any characters as specified by IDCAMS parameter definitions.
5. Replace 'mmm mmm' with the expected block size or control interval size. The default is 4088.

If you need a different block size, change the BLKSIZE entry in the QFILE Parameter of the PARMBLK.

6. In LBREST2, ensure that BUFFERSIZE is exactly twice the size of the control interval.

Note: For more information on how to calculate the VSAM space (records), the number of blocks (records), and code PARMBLK parameters, see the *Advantage VISION:Inform Installation Guide*.

Generated Output

The Library Restore Utility generates the following informational listings:

RESTBG.M4LIST	This listing summarizes the results of the background library restore process. It contains a list of the members that were restored.
RESTFG.INFPRINT	This listing summarizes the results of the foreground library restore process. It contains a list of the members that were restored, as well as the library space statistics (used blocks, free blocks, and so on).

Space Utilization Statistics

Each time you run the VISION:Inform Library Backup Utility against the foreground library, the utility generates space utilization statistics and writes data to the INFPRINT summary listing.

A sample of the actual message text follows. The asterisks in the message text indicate where the utility inserts numerical values for each run.

```
**S801 Of ******* overflow blocks, ******* are free, and ******* are in use.
**S802 Of ******* root blocks, ******* are free, and ******* are in use.
**S803 Of ******* root blocks, ******* have overflowed.
```

With these space utilization statistics, the system administrator can now determine:

- When to allocate additional space for the library (usually when the number of available overflow blocks becomes too small).
- When to increase the number of root blocks (usually when the number of available root blocks becomes too small).
- When to increase the block sizes (usually when the number of overflow blocks becomes too high).
- When to increase the bucket sizes (usually when the number of overflow blocks becomes too high).

Chapter 13 Promote Process Utility

This chapter describes the JCL, input control statement requirements, and options for executing the Promote Process Utility.

The VISION:Inform Promote Process Utility maintains the file, table, logical data view, and request (procedure) definitions in the background and foreground libraries.

- The utility's purpose is to migrate or promote definition information among the VISION:Inform libraries.
- The utility is a REXX EXEC command procedure that performs a series of functions to migrate or promote VISION:Inform definition library items to the VISION:Inform background and foreground libraries.
- The VISION:Inform Promote Process Utility runs as a batch job.

Promote Process Utility JCL

You build the JCL and control statements using the VISION:Workbench™ for ISPF Definition Processor Main Menu Promote option (Option 31).

- Through a series entries in panels and dialog interactions, you specify all information required for the Promote Process Utility.
- The utility uses your entries to create the member name and the JCL stream for the Promote process job. The utility saves the member using the name you specify.
- When you run the Promote process job, the Definition Processor migrates (promotes) the selected definition items to the background and foreground libraries.

Process Components

The following overviews the major components of the Promote process:

- The REXX command procedure, which is the Promote Process Utility, can be compared to a multiple-step job stream.
- Control statements and parameters drive the utility job, which executes many programs in a series, processing and manipulating data, and updating appropriate libraries.
- The utility also produces output logs that document actions and flow. Output listings display information about the processed items.

VISION:Inform Libraries

The primary purpose of the Promote Process Utility is to migrate definition information among the libraries used by VISION:Inform.

Definition Library	The definition library contains the source format versions of the definition items used in VISION:Inform. These items are file definitions, table definitions, logical data view definitions, and procedures. You can easily develop and maintain the source format details using the VISION:Workbench for ISPF Definition Processor.
Background Library	The background library, an execution time library, contains the items the VISION:Inform Background Processor uses while processing the various queries and tasks submitted by the user. The Promote process transforms items from source format to execution format during the migration into the background library.
Foreground Library	The foreground library, an execution time library, contains the items the VISION:Inform Foreground Processor uses. The items in the foreground library are grouped into two basic categories, definitions and user profiles.

- The definitions are the items migrated from the background library and transformed into online format.
- The user profiles are entered by the system administrator using the Foreground Processor. The user profiles define system security and access control for the online portion of VISION:Inform.

Flow Diagram

The following Promote Process Utility diagram shows the elements and how the flow of information occurs.

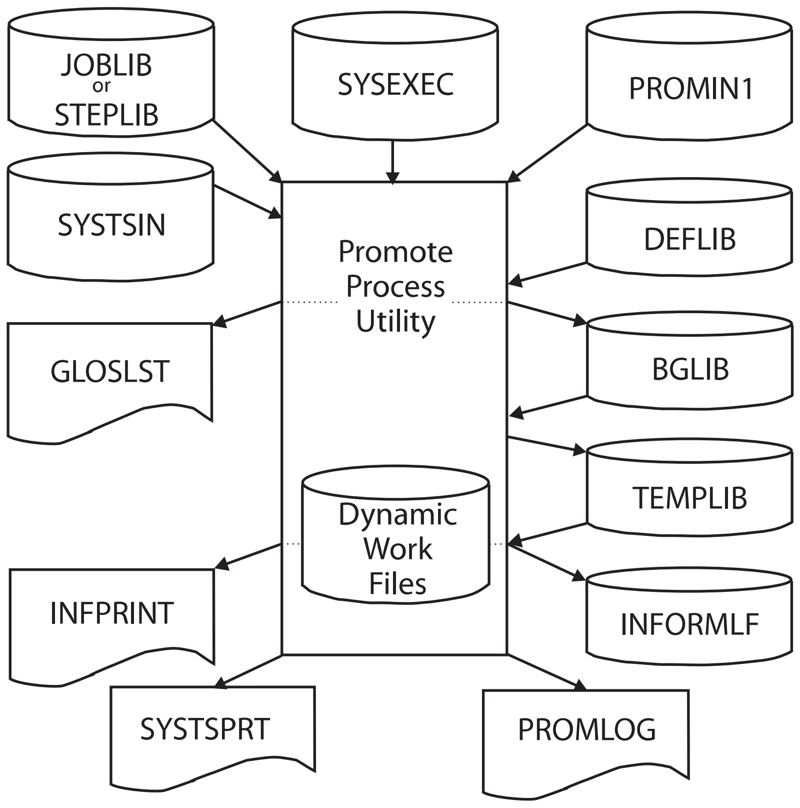

Figure 13-1 Promote Process Utility Flow Diagram

Executing the Utility

Note: You create the member PROMOTE by running the Promote process. The member PROMOTE is not delivered with INFORM.JCL.

The Promote Process Utility is a REXX EXEC command procedure that you run as a batch job by running the Definition Processor Promote option (Option 31):

```
INFORM.JCL(PROMOTE)
```

Specifying the Execution JCL

The following figure shows the sample REXX EXEC command procedure contained in INFORM.JCL in the member PROMOTE. Review the REXX EXEC carefully.

At minimum, make the following changes before submitting this job:

1. Supply your JOB card as the first statement.
2. Tailor the JCL to your installation's naming conventions and your specific requirements.
3. Specify the procedure variables—for information, see Specifying Procedure Variables.
4. Make additional modifications, as necessary, to conform to your installation's standards.

The following figure shows the sample JCL to run the Promote Process Utility:

```
//          JOB (
//*
//* THIS IS A SAMPLE CICS VISION:INFORM PROMOTE JOB STREAM
//*
//*                                     INFORM LOADLIB
//JOBLIB    DD  DSN=INFORM.LOADLIB,DISP=SHR
//*                                     IBM LANGUAGE ENVIRON LIB
//*
//*         DD  DSN=IBM.LANGUAGE.ENVR.RUNLIB,DISP=SHR
//*
//*
//*  THE PRIMARY PROMOTE STEP FOR CICS AND IMS ENVIRONMENTS
//*
//*                                     TSO ENVIRONMENT PROGRAM
//PROMOTE EXEC PGM=IKJEFT01,REGION=3072K
//*
//*                                     INFORM CLIST LIBRARY
//SYSEXEC   DD  DSN=INFORM.CLIST,DISP=SHR
//*                                     TSO ENVR PROG MSG LIST
//SYSTSPRT DD  SYSOUT=*
//*                                     TSO ENVR PROG INPUT STMT
//SYSTSIN   DD  *
  %M9JKP20
/*
//*                                     DEFINITION LIBRARY
//DEFLIB    DD  DSN=INFORM.DEFLIB,DISP=SHR
//*                                     BACKGROUND LIBRARY
//BGLIB     DD  DSN=INFORM.BGLIB,DISP=SHR
//*                                     FOREGROUND LIBRARY
//INFORMLF DD  DSN=INFORM.FGLIB,DISP=SHR
//*                                     TEMPORARY BGLIB
//TEMPLIB   DD  DSN=&&TEMPLIB,DISP=(,PASS),
//              SPACE=(TRK,0300,,CONTIG),UNIT=SYSDA
//*                                     WORKFILE FOR BGLIB ITEMS
//BGWORK    DD  DSN=&&WORK1,DISP=(,PASS),
//              SPACE=(TRK,(0300,30)),UNIT=SYSDA
//*                                     WORKFILE FOR FGLIB ITEMS
//FGWORK    DD  DSN=&&INFIN,DISP=(,PASS),
//              DCB=(LRECL=80,RECFM=F),UNIT=SYSDA,
//              SPACE=(TRK,(5,1))
//*                                     PROMOTE RUN LOG LISTING
//PROMLOG   DD  SYSOUT=*,
//              DCB=(RECFM=FA,LRECL=133)
//*                                     GLOSSARY LISTINGS
//GLOSLST   DD  SYSOUT=*,
```

Figure 13-2 Sample Execution JCL for the Promote Process Utility, PROMOTE (Page 1 of 2)

```
//             DCB=(RECFM=FA,LRECL=133)
//*                                        FOREGROUND PROMOTE LISTING
//INFPRINT DD  SYSOUT=*
//*                                        PROMOTE RUN CONTROL STMTS
//PROMIN1  DD  *
$RUNTYPE PROMOTE,CICS
$PARAMS  GLOSSARY=YES,CONDENSE=YES,ITEMS=SELECT,PROCEDURES=YES
$FILE    NAME=
/*
```

Figure 13-2 Sample Execution JCL for the Promote Process Utility, PROMOTE (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the variables you need to specify when executing the Promote Process Utility procedure, PROMOTE.
Note: Some of the processing programs called by the Promote Process Utility use the IBM Language Environment (known as LE or LE/370) Runtime Library Routines.

You must make the LE Runtime Library available to JOBLIB or STEPLIB for the Promote Process Utility.

The EXEC statement specifies the TSO environment program (terminal monitor program) that executes REXX EXECs and CLISTs in a background processing run.

Variable Name	Description
JOBLIB or STEPLIB	Specifies the VISION:Inform installation load library.
SYSEXEC	Specifies the VISION:Inform CLIST library that contains the CLISTs used by VISION:Workbench for ISPF Definition Processor. This is where the utility resides.
SYSTSIN	Specifies the REXX EXEC (the Promote Process Utility) name to be executed. The DD and input statement must be coded as shown.
PROMIN1	Specifies the run control and parameter input statements that indicate the utility run type, the actions to be performed, and items to be used. These statements are described later in this chapter.
DEFLIB	Specifies the definition library that contains the source format items to be promoted.
BGLIB	Specifies the background library that will contain the execution format items promoted for use by the Background Processor.
TEMPLIB	Specifies a temporary work data set that the Promote Process Utility uses. This space is for intermediate staging of the background library. ■ The space is allocated in contiguous tracks on SYSDA and is deleted at the end of the utility run. ■ The standard default value is 300 tracks. ■ You can specify a larger value if needed for your background library.
INFORMLF	Specifies the foreground library that contains the execution format items promoted for use by the Foreground Processor.
Dynamic Work Files	The utility dynamically allocates and deletes work files during execution. These temporary disk data sets are allocated to SYSDA.
GLOSLST	Specifies the destination of the glossary listings that the utility run produces. ■ Specify SYSOUT or a data set. ■ The minimum DCB requirements are RECFM FA or FBA, LRECL=133.

Variable Name	Description
INFPRINT	Specifies the destination of the log that the foreground Promote function within the utility produces. Specify SYSOUT or a data set.
SYSTSPRT	Specifies the destination of the TSO Terminal Monitor Program information and messages listing. Specify SYSOUT or a data set.
PROMLOG	Specifies the destination of the Promote Process Utility log listings. The output is a log of actions performed by the utility supplemented with related information and messages. ■ Specify SYSOUT or a data set. ■ The minimum DCB requirements are RECFM FA or FBA, LRECL=133.

Figure 13-3 JCL Procedure Variables for the Promote Process Utility

Specifying Control Statements

The Promote Process Utility uses the following control statements as input:

$RUNTYPE	Indicates the type of utility run; test only or an actual Promote run.
$PARAMS	Establishes the utility run options and actions.
$TABLE	Names specific definition library table definitions to promote.
$FILE	Names specific definition library file definitions to promote.
$LDV	Names specific definition library logical data view definitions to promote.

Coding Rules

Use the following rules when writing the Promote Process Utility control statements.

- Each control statement consists of a statement name and one or more keyword parameters.
- Start the name of each control statement in column 1.
- Start the keyword parameters in column 10 and separate them by commas or blanks.
- Specify the keyword parameters in any order.
- Control statements cannot be continued; all entries must be on one statement.

$RUNTYPE Control Statement

$RUNTYPE { PROMOTE | TESTRUN }

You must include the $RUNTYPE control statement and make it the first control statement input to the utility.

The $RUNTYPE statement parameter specifies the type of run as PROMOTE or TESTRUN.

PROMOTE	Perform the Promote process. The background and foreground libraries are updated. The run actions and item actions are performed as specified. Error checking is performed.
TESTRUN	Perform a test of the Promote process. The background and foreground libraries are not updated. The specified run and item actions are simulated and error checking is performed.

Note: If you are using the VISION:Inform Definition Processor Main Menu Promote option, the Promote process job run types are REAL (equivalent to the utility PROMOTE) and TEST (equivalent to the utility TESTRUN).

$PARAMS Control Statement

$PARAMS GLOSSARY={ YES | <u>NO</u> | NONE }, CONDENSE={ YES | <u>NO</u> },
ITEMS={ <u>ALL</u> | SELECT | NONE }, PROCEDURES={ YES | <u>NO</u> }

The $PARAMS control statement is optional. If omitted, standard defaults will be used.

$PARAMS control statement parameters:

GLOSSARY	Specify YES to print a glossary listing for every item selected from the definition library to be promoted to the background and foreground libraries.
	Specify NO or NONE to suppress the glossary listing.
	You can override this entry for specific items named on the $TABLE, $FILE, and $LDV control statements.
CONDENSE	Specify YES to condense the background library after the items from the definition library are promoted.
	Specify NO to suppress condensing the background library.
	Condensing the background library recovers the space from deleted items.
ITEMS	Specify ALL to promote all the definitions in the background library to the foreground library.
	Specify SELECT to promote only the selected definitions (named in the $TABLE, $FILE, and $LDV control statements) to the foreground library. If you specify SELECT, you must include a $TABLE, a $FILE, or a $LDV statement.
	Specify NONE to suppress promoting any definitions from the background library to the foreground library.

$TABLE, FILE, and, $LDV Control Statements

$TABLE NAME=*nnnnnnnn*
$FILE GLOSSARY={ YES | NO | NONE | BYNAME | BYLOCATION },
$LDV DELETE=ITEM

These control statements name the specific items to be promoted from the definition library to the background library. Use $TABLE to identify table definitions, use $FILE to identify file definitions, and use $LDV to identify logical data view definitions.

- If you specify ITEMS=ALL in the $PARAMS control statement, do not include a $TABLE, $FILE, or $LDV statement.
- If you specify ITEMS=SELECT in the $PARAMS control statement, specify a $TABLE, $FILE, or $LDV statement.

Only the selected items are promoted from the background library to the foreground library.

$TABLE, $FILE, and $LDV control statement parameters:

NAME	Identifies the item and member in the definition library.
GLOSSARY	This entry overrides the $PARAMS GLOSSARY entry for the specific item named on the $TABLE, $FILE, or $LDV control statement. Specify YES to print a glossary listing for this item. Specify NO or NONE to suppress the printing of the glossary listing. Specify BYNAME or BYLOCATION to print a glossary listing for the item in order by field name or in order by field location, respectively. These keywords only apply to file definition items. For other items types, these entries are interpreted as a YES entry.
DELETE	Specify ITEM to indicate that the named item is to be deleted from background and foreground libraries. The item/member in the definition library is not altered.

Generated Output

The Promote Process Utility produces the following output:

GLOSLST	The glossary listing shows the specific detail information about items that the utility produces.
INFPRINT	The foreground promote log shows the actions that took place at foreground promote function within the utility run.
SYSTSPRT	The TSO Terminal Monitor Program information and messages listing.
PROMLOG	Review this log to ensure that the desired actions occurred as expected and to see if any inconsistencies were encountered.

Chapter 14

Query Migration Utility (LIBCOPY)

The Query Migration Utility, LIBCOPY, extracts items of type QUERY and type STMTS from one VISION:Inform system foreground library and then migrates them to a second VISION:Inform system foreground library.

Note: Take the VISION:Inform target foreground library offline when running this utility.

This utility does not address user profile requirements for the items being extracted. This means, if you copy a query to a user profile and the database the query is for is not in the user profile being copied to, or has a different set of restrictions than that of the “from” profile, you will need to modify these items manually when updating the affected profiles.

Utility Component

The Query Migration Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(LIBCOPY)

This sample JCL to execute the Migration Utility is in the VISION:Inform JCL library, member LIBCOPY.

Flow Diagram

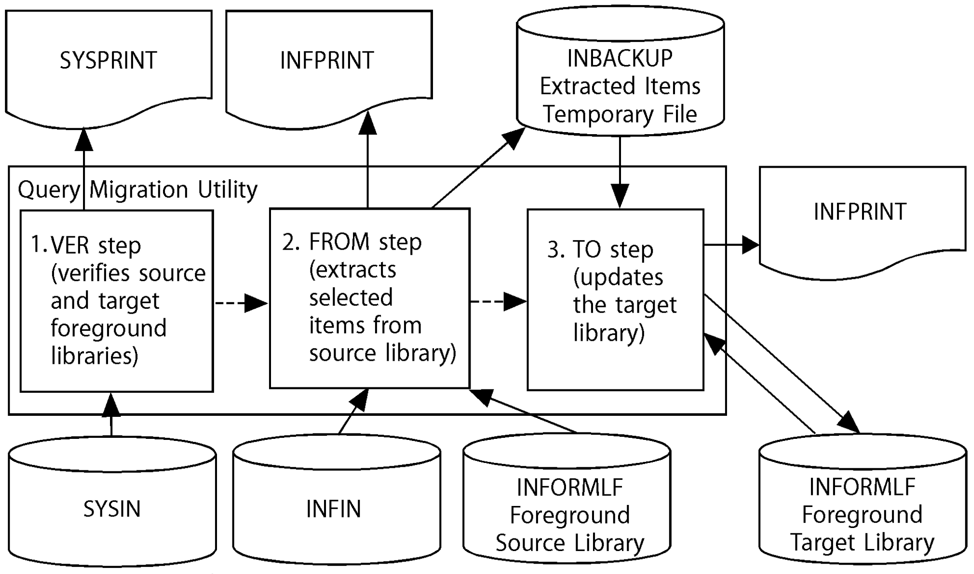
SYSPRINT
INFPRINT
INBACKUP Extracted Items Temporary File
Query Migration Utility
1. VER step (verifies source and target foreground libraries)
2. FROM step (extracts selected items from source library)
3. TO step (updates the target library)
INFPRINT
SYSIN
INFIN
INFORMLF Foreground Source Library
INFORMLF Foreground Target Library

Figure 14-1 Query Migration Utility Flow Diagram

Note: The step names correspond to the JCL job step names.

Utility Flow

The Query Migration Utility, LIBCOPY, consists of the following steps:

Note: Take the VISION:Inform target foreground library offline when running this utility.

Note: The step names correspond to JCL job step names.

1. VER step

   This step executes the VSAM IDCAMS utility to verify that all the VSAM files are offline and available to the migration job. The following files are verified:

   – VISION:Inform source foreground library.

   – VISION:Inform target foreground library.

   The utility creates the SYSOUT data set, VER.SYSPRINT.

   The acceptable return code for this step is 0.

2. FROM step

   This step builds a temporary sequential file containing the items selected from the source library based upon valid COPY control statements. This temporary file becomes the input file to the next step.

   The utility creates the SYSOUT data set, FROM.INFPRINT.

The acceptable return codes for this step are 0, 4, or 8. For more information on return codes, see the *Advantage VISION:Inform Messages and Codes* manual.

3. TO step

   This step updates the target foreground library with the selected items from the temporary file created in the previous step.

   The utility creates the SYSOUT data set, TO.INFPRINT.

   The acceptable return codes are 0 or 4. For more information on return codes, see*Advantage VISION:Inform Messages and Codes* manual.

Executing the Utility

The LIBCOPY utility is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(LIBCOPY)

This JCL contains an instream procedure, followed by the steps to execute this procedure.

Specifying the Execution JCL

Figure 14-2 shows the sample JCL contained in INFORM.JCL in the member LIBCOPY. Review this JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Modify all specifications shown in lowercase letters.
3. Supply procedure variable values—for information, see Specifying Procedure Variables.
4. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the Query Migration Utility.

```
//* MEMBER LIBCOPY
//**********************************************************************
//*   THIS PROCEDURE COPIES FOREGROUND LIBRARY ITEMS OF TYPE QUERY OR *
//*   STMTS FROM ONE FOREGROUND LIBRARY TO ANOTHER.                   *
//**********************************************************************
//LBCOPY   PROC FROMLIB=,
//            TOLIB=,
//            FROMLOAD=,
//            TOLOAD=,
//            COND=,
//            RGN=
//VER    EXEC PGM=IDCAMS,REGION=&RGN
//TOLIB    DD DISP=SHR,DSN=&TOLIB
//FROMLIB  DD DISP=SHR,DSN=&FROMLIB
//SYSPRINT DD SYSOUT=*
//FROM   EXEC PGM=INFORMUX,REGION=&RGN
```

Figure 14-2 Sample Execution JCL for the Query Migration Utility, LIBCOPY (Page 1 of 2)

```
//STEPLIB  DD DISP=SHR,DSN=&FROMLOAD
//INFPRINT DD SYSOUT=*
//INFORMLF DD DISP=SHR,DSN=&FROMLIB
//INBACKUP DD DISP=(NEW,PASS),DSN=&&COPY,UNIT=SYSDA,
//            SPACE=(TRK,(20,20))
//TO      EXEC PGM=INFORMUY,REGION=&RGN,COND=&COND
//STEPLIB  DD DISP=SHR,DSN=&TOLOAD
//INFPRINT DD SYSOUT=*
//INFORMLF DD DISP=SHR,DSN=&TOLIB
//INBACKUP DD DISP=OLD,DSN=&&COPY
//       PEND
//*******************************************************************
//* THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.          *
//* BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                         *
//*                                                                 *
//* FROMLIB  - THE SOURCE INFORM FOREGROUND LIBRARY.                *
//* TOLIB    - THE TARGET INFORM FOREGROUND LIBRARY.                *
//* FROMLOAD - THE SOURCE INFORM INSTALLATION LOAD LIBRARY.         *
//* TOLOAD   - THE TARGET INFORM INSTALLATION LOAD LIBRARY.         *
//* COND     - CONDITION CODE FOR EXECUTION OF "TO" STEP. THE       *
//*            DEFAULT IS (0,NE).                                   *
//* RGN      - REGION SIZE. DEFAULT IS 1M.                          *
//*                                                                 *
//* YOU MUST ALSO SPECIFY THE ITEMS TO BE COPIED, THE NAMES TO BE   *
//* USED IN THE TARGET FOREGROUND LIBRARY, AND OPTIONALLY SPECIFY   *
//* THE REPLACE OPTION, AFTER THE "FROM.INFIN" DD STATEMENT.        *
//*******************************************************************
//COPY    EXEC LBCOPY,
//             FROMLIB='INFORM.TEST.FGLIB',
//             TOLIB='INFORM.PROD.FGLIB',
//             FROMLOAD='INFORM.TEST.LOADLIB',
//             TOLOAD='INFORM.PROD.LOADLIBB',
//             COND='(0,NE)',
//             RGN=1M
//VER.SYSIN DD *
         VERIFY FILE(FROMLIB)
         VERIFY FILE(TOLIB)
//FROM.INFIN DD *
  COPY name type userid TO name type userid
  COPY name type userid TO name type userid REPLACE
```

Figure 14-2 Sample Execution JCL for the Query Migration Utility, LIBCOPY (Page 2 of 2)

Specifying Procedure Variables

The following summarizes the variables you need to specify when executing the Query Migration Utility JCL procedure, LIBCOPY.

Variable name	Description
FROMLIB	Specify the name of the foreground library that is the source of the items you specify in the COPY control statements.
TOLIB	Specify the name of the foreground library that will be updated with the items extracted from the source foreground library.
FRMLOAD	Specify the name of the VISION:Inform installation load library to be used by the FROM step.
TOLOAD	Specify the name of the VISION:Inform installation load library to be used by the TO step.

Figure 14-3 JCL Procedure Variables for the LIBCOPY Utility

Specifying the COPY Control Statement

COPY *name*1 *type*1 *userid*1 TO *name*2 *type*2 *userid*2 {REPLACE}

The required COPY control statement specifies the names of items (of type QUERY and type STMTS) to be copied from a source foreground library and the corresponding names in the target foreground library.

You can process multiple COPY statements in a single run.

COPY statement parameters:

COPY	Required. COPY is a control statement command that identifies the control statement type.
name1	The name entry for the COPY keyword supports a simple wild card construct adhering to the following rules: ■ The wild card character is a single asterisk (*). ■ Specify only one wild card character in a name specification. ■ If you specify a wild card, make it the last or only character in the name field of the COPY and TO keywords. ■ If you use a wild card with other characters, use the same the character count of the name specifications for both the COPY and TO name keywords.
type1	Specify as QUERY or STMTS.
userid1	Specify a valid VISION:Inform user ID from 1 to 8 characters.
TO	TO is a required keyword.
name2	Specify TO keyword name2 with the same characteristics as name1.
type2	Specify QUERY or STMTS.
userid2	Specify a valid VISION:Inform user ID from 1 to 8 characters.
REPLACE	Optional. ■ If you specify REPLACE, place it in the same relative position as shown in COPY statement format (above). ■ If you do not specify REPLACE and the TO item exists, the COPY operation does not take place.

The COPY statements you specify are input to the Query Migration Utility, LIBCOPY.

Generated Output

The Query Migration Utility creates validation listings for you to review.

FROM.INFPRINT	To verify that the items specified in the COPY control statements have been successfully retrieved and written to the temporary data set, review the migration listing that is written to the FROM.INFPRINT SYSOUT data set. You can find error messages, informational messages, and explanations in the *Advantage VISION:Inform Messages and Codes* manual.
TO.INFPRINT	To verify that the target library was successfully updated with at least some of the extracted item, you must review the migration listing that is written to the TO.INFPRINT SYSOUT data set. You can find error messages, informational messages, and explanations in the*Advantage VISION:Inform Messages and Codes* manual.

Chapter 15 VISION:Builder Quick Start Utility (BUILDRQS)

The VISION:Builder Quick Start Utility, BUILDRQS, populates a VISION:Inform definition library with information retrieved from the VISION:Inform background library or the VISION:Builder COMLIB common library. This utility retrieves promoted definitions from the VISION:Inform background library or the VISION:Builder COMLIB common library for use by the Definition Processor.

The previous name for the VISION:Builder Quick Start Utility, BUILDRQS, was the Source Statement Retrieval Utility, SSR.

Utility Component

The VISION:Builder Quick Start Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(BUILDRQS)

This sample JCL to execute the VISION:Builder Quick Start Utility is in the VISION:Inform JCL library, member BUILDRQS.

Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

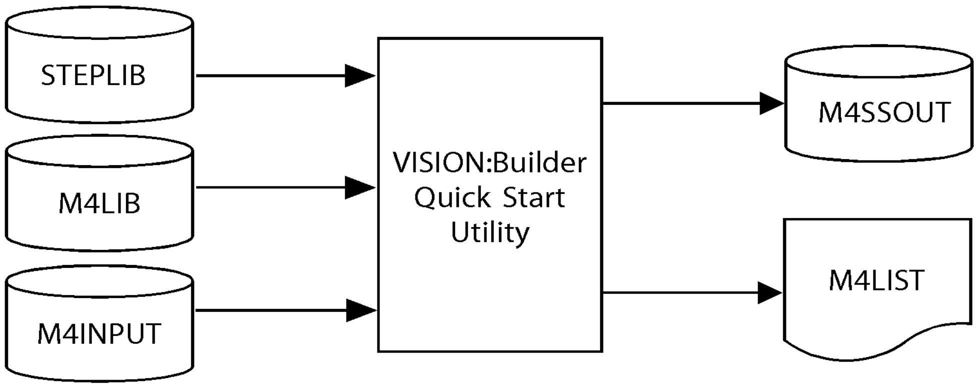

Figure 15-1 VISION:Builder Quick Start Utility Flow Diagram

The VISION:Builder Quick Start Utility uses the following three input data sets:

STEPLIB	Specifies the VISION:Inform installation load library.
M4LIB	Specifies the background library from which the source will be retrieved.
M4INPUT	Specifies the appropriate source retrieval control statements. See Providing DD Statement Overrides.

The VISION:Builder Quick Start Utility creates the following output files:

M4SSOUT	Specifies the definition library to which the generated source statements will be written.
M4LIST	Specifies a SYSOUT file. A listing of the M4INPUT source retrieval control statements is written to this data set.

Executing the Utility

The VISION:Builder Quick Start Utility, BUILDRQS, is a batch utility you run by submitting the JCL provided in:

INFORM.JCL(BUILDRQS)

This JCL contains an instream procedure, followed by the step to execute the procedure.

Specifying the Execution JCL

Figure 15-2 shows the sample JCL contained in INFORM.JCL in the member BUIDRQS. Review the supplied JCL carefully.

Note: You can also run this utility interactively under TSO/ISPF using the VISION:Inform Definition Processor Import option.

For additional information, see the *Advantage VISION:Inform Definition Processor Reference Guide*.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply values for the procedure variables—for information, see Following is the JCL to run the VISION:Builder Qucik Start Utility..
3. Complete the DD statement override for SSR.M4INPUT. See Providing DD Statement Overrides.
4. Make additional modifications, as needed, to conform to your installation's standards.

Following is the JCL to run the VISION:Builder Qucik Start Utility.

```
//* MEMBER BUILDRQS                                                     00010000
//********************************************************************* 00020003
//* PROCEDURE TO RETRIEVE DEFINITION SOURCE STATEMENTS FROM A         * 00030003
//* BACKGROUND LIBRARY, AND STORE THEM AS A MEMBER IN THE DEFINITION  * 00040003
//* LIBRARY.                                                          * 00050003
//* *** NOTE ***  IF THE LANGUAGE ENVIRONMENT LOAD LIBRARY IS NOT IN  * 00060003
//*               THE STANDARD LOAD LIBRARY LIST MADE AVAILABLE TO    * 00070003
//*               ALL BATCH PROGRAMS ON YOUR HOST SYSTEM, IT MUST BE  * 00080003
//*               CONCATENATED TO THE SSR.STEPLIB DD STATEMENTS.      * 00090003
//********************************************************************* 00100003
//SSR     PROC LOADLIB=,                                                00110000
//            BGLIB=,                                                   00120001
//            DEFLIB=,                                                  00130001
//            MEMBER=                                                   00140001
//SSR    EXEC PGM=MARKIV,REGION=2048K                                   00150001
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00160002
//M4LIST   DD SYSOUT=*                                                  00170001
//M4LIB    DD DISP=SHR,DSN=&BGLIB                                       00180001
//M4SSOUT  DD DISP=OLD,DSN=&DEFLIB(&MEMBER)                             00190002
//M4INPUT  DD DUMMY                                                     00200001
//       PEND                                                           00210001
//********************************************************************* 00220003
//*  THE FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE.           * 00230003
//*  BEFORE YOU RUN THIS PROCEDURE, SPECIFY:                          * 00240003
//*                                                                   * 00250003
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                * 00260003
//*    BGLIB   - THE INFORM BACKGROUND LIBRARY.                       * 00270003
//*    DEFLIB  - THE INFORM DEFINITION LIBRARY.                       * 00280003
//*    MEMBER  - MEMBER NAME FOR THE DEFINITION YOU ARE GENERATING.   * 00290003
//*                                                                   * 00300003
//*  YOU MUST ALSO PROVIDE THE APPROPRIATE SYSIN DATA IN THE          * 00310003
//*  SSR.M4INPUT DD OVERRIDE STMT.                                    * 00320003
```

Figure 15-2 Sample Execution JCL for VISION:Builder Quick Start Utility, BUILDERQS (Page 1 of 2)

```
// ***************************************************************** 00330003
//GETSRC  EXEC SSR,                                                     00340000
//             LOADLIB='INFORM.LOADLIB',                                00350001
//             BGLIB='INFORM.BGLIB',                                    00360001
//             DEFLIB='INFORM.DEFLIB',                                  00370001
//             MEMBER='DEFNAME'                                         00380000
//SSR.M4INPUT DD *                                                      00390000
SSR     RC                    B      S                                  00400000
        CTOPRDEFNAME                                                    00410000
```

Figure 15-2 Sample Execution JCL for VISION:Builder Quick Start Utility, BUILDERQS (Page 2 of 2)

Specifying Procedure Variables

The following table summarizes the variables you need to specify when executing the VISION:Builder Quick Start Utility JCL procedure, BUILDRQS.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
BGLIB	Specify the name of the background library from which the source is to be retrieved.
DEFLIB	Specify the definition library to which the new definition source file will be written.
MEMBER	Specify a member name for the new definition. Use the same name that is specified in the M4INPUT DD statement override information for "defname." ■ If you specify an existing member name, the utility replaces it. ■ If you specify a new name, the utility adds it.

Figure 15-3 JCL Procedure Variables for the VISION:Builder Quick Start Utility

Providing DD Statement Overrides

Provide the following DD statement overrides when executing the VISION:Builder Quick Start Utility.

In the following examples, you modify the text in lowercase letters. Do not change the text in uppercase letters.

//SSR.M4INPUT

The following figure shows the SSRM4INPUT sample input.

```
....+....+....+....+....+....+....+....+....+
//SSR.M4INPUT DD  *
SSR     RC                    B      S
        CToprdefname
```

Figure 15-4 SSR.M4INPUT Sample Input

Specify the source generation control statements in the M4INPUT file.

1. Replace 'opr' with one of the following source generation operators.

   RTD — Retrieve table definition source.

   RFD — Retrieve file definition source.

   RDB — Retrieve logical data view source.

   RRG — Retrieve request and procedure source.

   Put the operator in columns 11-13.
2. Replace 'defname' with the name of the source member that you want to create in the specified definition library. Match the name with the member name specified in the procedure variable MEMBER.

   If you specify the member name, place it in columns 14-21.

Generated Output

The VISION:Builder Quick Start Utility writes a list of the M4INPUT statements to the M4LIST file.

Chapter 16
VISION:Inquiry Quick Start Utility (INQRYQS)

The VISION:Inquiry Quick Start Utility, INQRYQS, generates VISION:Inform file definitions from existing VISION:Inquiry MAPGENs. During the utility processing, each VISION:Inquiry map that is processed becomes a separate VISION:Inform file definition. The VISION:Inquiry map information is converted into file, segment, and field information in the file definition.

Utility Component

The VISION:Inquiry Quick Start Utility is composed of the following component:

Sample Execution JCL — INFORM.JCL(INQRYQS)

This sample JCL to execute the VISION:Inquiry Quick Start Utility is in the VISION:Inform JCL library, member INQRYQS.

Running the Utility

You can run the VISION:Inquiry Quick Start Utility either from the Definition Processor Main Menu Option 19 (Import option) or as a stand-alone batch execution.

- For information on using the VISION:Inquiry Quick Start Utility from within the Definition Processor, see the *Advantage VISION:Inform Definition Processor Reference Guide*.
- This chapter describes the use of the VISION:Inquiry Quick Start Utility as a batch utility.

Generating Multiple File Definitions

You can generate multiple file definitions in a single VISION:Inquiry Quick Start Utility batch execution.

- The utility stores each generated file definition as a separate member in the indicated source definition library.
- The utility uses the specified file name as the member name when the generated definition is saved in the source definition library.

If you specify an existing member name, the utility replaces the existing member with the newly saved definition; otherwise it creates a new member name.

Editing Generated File Definitions

Once the utility creates a VISION:Inform file definition, you can add any additional information that is needed using the Definition Processor.

Flow Diagram

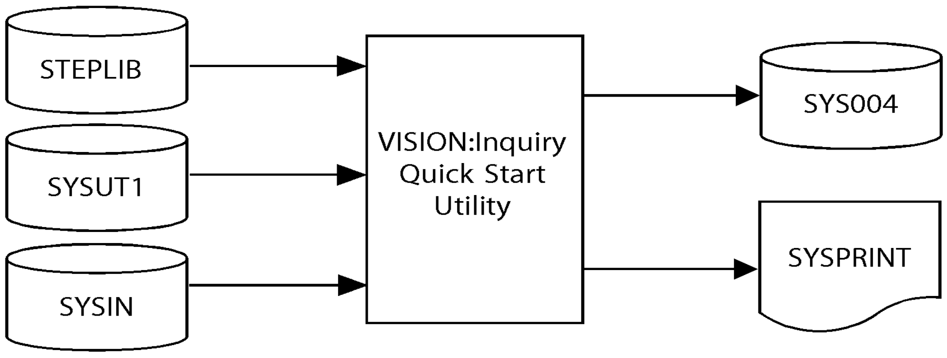

Figure 16-1 VISION:Inquiry Quick Start Utility Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

The VISION:Inquiry Quick Start Utility uses the following three input data sets:

STEPLIB	Specifies the VISION:Inform installation load library.
SYSIN	Specifies the VISION:Inquiry Quick Start Utility control statements.
SYSUT1	Contains the unloaded form of the VISION:Inquiry system database. See the *Advantage VISION:Inquiry Technical Reference Guide* for your environment, for information on how to create the unloaded form of the system database.

The VISION:Inquiry Quick Start Utility produces the following two output files:

SYSPRINT	Contains a summary report of the file definition generation process.
SYS004	The utility writes the generated file definitions to the partitioned data set pointed to by the DD name SYS004. This must be a partitioned data set that usually specifies the definition library.

Executing the Utility

The VISION:Inquiry Quick Start Utility is a batch utility you run by submitting the JCL contained in:

INFORM.JCL(INQRYQS)

This JCL contains an instream procedure followed by the step to execute the procedure, as well as sample control statements.

Note: For information on using the VISION:Inquiry Quick Start Utility from within the Definition Processor, see the *VISION:Inform Definition Processor User's Guide*.

Specifying the Execution JCL

Figure 16-2 shows the JCL contained in INFORM.JCL in the member INQRYQS. Review the JCL carefully.

At a minimum, make the following changes before submitting this JCL:

1. Supply a JOB card.
2. Supply values for the procedure variables—for information, see Specifying Procedure Variables.
3. Provide a DD statement override for INQRYQS.SYSIN. This data set contains the required VISION:Inquiry Quick Start input control statements which are used to control the generation of VISION:Inform file definitions.

   For detailed information on these control statements, see Specifying the FILEGEN Control Statement.
4. Make additional modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the VISION:Inquiry Quick Start Utility.

```
//* MEMBER INQRYQS                                                      00010000
//********************************************************************* 00020000
//* UTILITY TO CONVERT VISION:INQUIRY FILE DEFINITIONS INTO           * 00030000
//* VISION:INFORM FORMAT FILE DEFINITIONS.                            * 00040000
//* THE VISION:INQUIRY FILE DEFINITIONS MUST COME FROM A              * 00050000
//* VISION:INQUIRY UNLOADED SYSTEM DATABASE FILE.  SEE YOUR           * 00060000
//* VISION:INQUIRY TECHNICAL REFERENCE MANUAL FOR INFORMATION ON      * 00070000
//* HOW TO CREATE AN UNLOADED COPY OF THE SYSTEM DATABASE.            * 00080000
//*                                                                   * 00090000
//* THIS UTILITY MAY ALSO BE INVOKED INTERACTIVELY UNDER TSO/ISPF     * 00100000
//* USING THE DEFINITION PROCESSOR IMPORT FUNCTION.                   * 00110000
//********************************************************************* 00120000
//INQRYQS PROC RGN=2M,                                                  00130000
//             LOADLIB=,                                                00140000
//             ULSYSDB=,                                                00150000
//             DEFLIB=                                                  00160000
//INQRYQS EXEC PGM=INQRYQS,REGION=&RGN                                  00170000
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB                                    00180000
//SYSPRINT  DD SYSOUT=*                                                 00190000
//SYSUT1    DD DISP=SHR,DSN=&ULSYSDB                                    00200000
//SYS004    DD DISP=OLD,DSN=&DEFLIB                                     00210000
```

Figure 16-2 Sample Execution JCL for the VISION:Inquiry Quick Start Utility, INQRYQS (Page 1 of 2)

```
//        PEND                                                          00220000
//********************************************************************* 00230000
//*  FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU    * 00240000
//*  RUN THIS PROCEDURE, SPECIFY:                                     * 00250000
//*                                                                   * 00260000
//*    RGN     - THE REGION SIZE; DEFAULT IS 2M.                      * 00270000
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                * 00280000
//*    ULSYSDB - THE UNLOADED VISION:INQUIRY SYSTEM DATABASE FILE.    * 00290000
//*    DEFLIB  - THE INFORM DEFINITION LIBRARY.                       * 00300000
//********************************************************************* 00310000
//STEP01  EXEC INQRYQS,RGN=2M,                                          00320000
//             LOADLIB='INFORM.LOADLIB',                                00330000
//             ULSYSDB='VISION.INQUIRY.UNLOADED.SYSDBASE',              00340000
//             DEFLIB='INFORM.DEFLIB'                                   00350000
//SYSIN DD *                                                            00360000
  FILEGEN NAME=VSHPLANT,FLDPREFX=PLT                                    00370000
  FILEGEN NAME=SALARIES,FLDPREFX=SAL                                    00380000
```

Figure 16-2 Sample Execution JCL for the VISION:Inquiry Quick Start Utility, INQRYQS (Page 2 of 2)

If you specify the SYSIN data set as a DUMMY or if you do not specify any FILEGEN statements in the input stream (empty SYSIN data set), all MAPGENs in the unloaded system database will be converted in one call of the VISION:Inquiry Quick Start Utility.

Specifying Procedure Variables

The following summarizes the procedure variables you need to specify when executing the VISION:Inquiry Quick Start Utility JCL procedure, INQRYQS.

Variable Name	Description
LOADLIB	Specify the name of the VISION:Inform installation load library.
ULSYSDB	Specify the name of the VISION:Inquiry unloaded system database.
DEFLIB	Specify the name of the definition library.

Figure 16-3 JCL Procedure Variables for the VISION:Inquiry Quick Start Utility

Specifying the FILEGEN Control Statement

The VISION:Inquiry Quick Start Utility uses the following control statement as input:

FILEGEN	The FILEGEN statement starts a new file definition. This utility uses the NAME parameter specified on the FILEGEN statement both as the file name and the member name when the utility generates and saves the file definition.

Coding Rules

Apply the following rules when specifying the VISION:Inquiry Quick Start Utility control statements:

- Place each statement on a single line and begin the statement with the FILEGEN command name.
- Each control statement can contain keyword parameters.
  - You can specify keyword parameters in any order.
  - Separate the parameters by commas.
  - Do not use embedded blanks between parameters.
- The NAME keyword is required on each statement. All other keywords are optional.
- Do not use blank lines within the SYSIN file.
- The utility scans columns 1 through 71 of the control statement and ignores columns 72 through 80.

FILEGEN Control Statement

FILEGEN NAME=filenam, BUFFSIZE=nnnn, FLDPREFIX=pre, NEWNAME=newname

FILEGEN	Required. FILEGEN is a control statement command. It identifies the control statement type. The FILEGEN control statement starts a new file definition.
NAME	Required. The NAME parameter specifies the name of the MAPGEN that is being converted. Specify from 1 to 8 characters for the map name. If the utility cannot find the corresponding MAPGEN in the unloaded VISION:Inquiry system database, the utility prints a message indicating that no conversion took place. When the utility saves the generated file definition, the utility uses file name as the member name unless you specify the NEWNAME keyword. If you specify an existing member name, the utility replaces the existing member with the newly saved definition; otherwise, the utility creates a new member name.

BUFFSIZE	Optional. The BUFFSIZE parameter defines the size of the buffer needed to process a logical record for IMS. ■ Enter a number from 1 to 32760. ■ Enter multiples of 1024 as "nnnnK" where "nnnn" is a number between 1 to 9999. Use this parameter only for IMS definitions. The utility ignores this parameter for all other file definition types.
FLDPREFX	Optional. The FLDPREFX parameter specifies the prefix for generating primary field names in the VISION:Inform file definition. ■ Specify from 1 to 3 characters for the prefix. Primary field names are required, must be unique, and must contain from 1 to 8 characters. ■ Since VISION:Inquiry field names can be longer than eight characters, the VISION:Inquiry Quick Start Utility automatically generates a unique 8-character primary field using the FLDPREFX value followed by a generated field number. ■ If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format Fnnnnnnn where nnnnnnn is a number from 0000001 to 9999999. For more information, see Editing the Generated VISION:Inform File Definition.
NEWNAME	Optional. The NEWNAME parameter specifies the name of the converted file definition. Specify from 1 to 8 characters for the name. ■ This utility uses this name as both the file definition name and the member name in the definition library. ■ If you omit this keyword, the utility uses the existing name, as specified in the NAME keyword, as the file definition name and member name.

Conversion Rules

The VISION:Inquiry Quick Start Utility generates a VISION:Inform file definition and field names that you can edit using the Definition Processor.

Editing the Generated VISION:Inform File Definition

After you edit a file definition, validate it prior to using it with the Definition Processor. Consider the following items when editing a generated file definition.

File Information	Add the glossary specification, as appropriate. Adjust the buffer size specification, as needed, for IMS files.
Segment Information	Provide additional segment information such as segment (row) order and suppress duplication, as needed. ■ Identify segment key fields during the conversion process based upon the information contained in the MAPGEN. ■ If a segment has more than one key field specified, adjust the key field specifications, as needed. ■ Identify segment count fields for VSAM hierarchical files, as appropriate.
Field Information	Modify primary field name assignments, if needed. Provide additional field information such as rounding, editing, column headings, and automatic table lookup results, as needed. The utility copies the field description information from the description information in the VISION:Inquiry MAPGEN. Since VISION:Inquiry MAPGENs do not contain column heading or equivalent information, no column heading information will be present in the converted file definition, but you can add it later.

Specifying VISION:Inform Field Names

Assign all fields within a VISION:Inform file definition a unique name. This name is referred to as the primary field name.

When you build a VISION:Inform file definition from a VISION:Inquiry MAPGEN, assign a unique primary field name to each field.

- Specify from 1 to 8 characters for the primary field name.
- Since VISION:Inquiry field names can be longer than eight characters, the VISION:Inquiry Quick Start Utility automatically generates a unique primary field name using the FLDPREFX parameter value on the FILEGEN statement followed by a generated field number.
- If you omit the FLDPREFX parameter, the default prefix is F and the generated primary field names have the format Fnnnnnnn where nnnnnnn is a number from 0000001 to 9999999. The original name is used as the alternate name, in this case.

Once you create the file definition, you can refine it using the Definition Processor. You can add column heading information and apply other changes, as appropriate.

Chapter 17 VISION:Results Quick Start Utility (RESULTQS)

The VISION:Results Quick Start Utility, RESULTQS, generates VISION:Inform file definitions from existing VISION:Results and VISION:Eighty file definitions.

- Input to the VISION:Results Quick Start Utility is in the form of a sequential data set, PDS (Partitioned Data Set) member, MVS copybook, CA-Panvalet copybook, or CA-Librarian copybook containing a single VISION:Results file definition.
- Output from the VISION:Results Quick Start Utility is in the form of a single PDS member containing the converted file definition in VISION:Inform format.

Utility Components

The VISION:Results Quick Start Utility is composed of the following components:

Sample Execution JCL — INFORM.JCL(RESULTQS)

This sample procedure to execute the VISION:Results Quick Start Utility is in the VISION:Inform JCL library as member RESULTQS.

Load Library — INFORM.LOADLIB

The installation load library contains the VISION:Results Quick Start Utility load module (RESULTQS) and various called modules (DYLPCPS, DYL280EF, DLY280LX, DYL280PX, and DYL280V).

Copy Interfaces — INFORM.WORKLIB

For optional support in accessing definitions stored in CA-Panvalet or CA-Librarian copybooks, two JCL members are supplied to link the appropriate interface routines with the VISION:Results Quick Start Utility. For more information see the *Advantage VISION:Inform Installation Guide for CICS*.

Flow Diagram

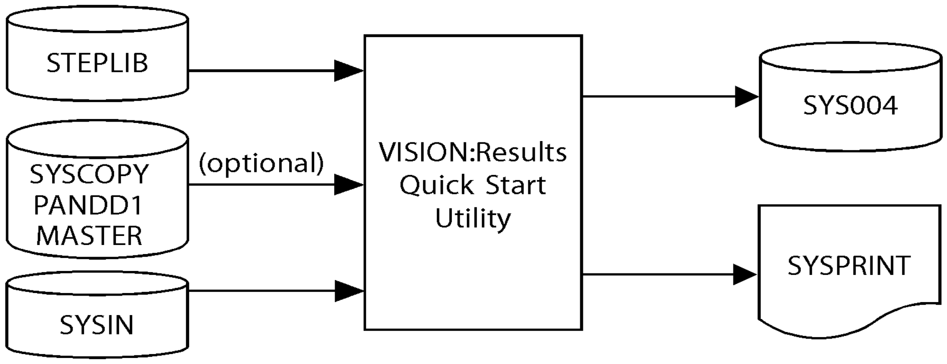

Figure 17-1 VISION:Results Quick Start Utility Flow Diagram

Note: Utility flow diagram names correspond to JCL DD names.

The VISION:Results Quick Start Utility uses the following input data sets:

STEPLIB	Required. Specify the VISION:Inform installation load library.
SYSIN	Required. Specify an input file to be converted as: ■ A member in a PDS (the default in the supplied JCL). ■ A sequential data set. ■ A VISION:Results COPY statement for data from an MVS copybook (SYSCOPY DD statement). ■ A VISION:Results COPYP statement for data from a CA-Panvalet library (PANDD1 DD statement). ■ A VISION:Results COPYL statement for data from CA-Librarian master file (MASTER DD statement). ■ As SYSIN instream data. For details on SYSIN data from sources other than a PDS, see Providing DD Statement Overrides. Specify input in a valid VISION:Results file definition beginning with a free-form FILE statement, followed by field definition statements.
SYSCOPY	Optional. If the SYSIN contains a COPY statement for a copybook from an MVS PDS member, then provide the SYSCOPY DD statement and point to the MVS PDS copy library.
PANDD1	Optional. If the SYSIN contains a COPYP statement for a copybook from a CA-Panvalet library, then provide the PANDD1 DD statement and point to the CA-Panvalet library.

MASTER	Optional. If the SYSIN contains a COPYL statement for a copybook from a CA-Librarian master file, then provide the MASTER DD statement and point to the CA-Librarian master file.

The VISION:Results Quick Start Utility produces the following output data sets:

SYSPRINT	The SYSPRINT data set contains a listing of the input statements, as well as any diagnostic or informational messages.
SYS004	The SYS004 file contains the converted file definition.

Executing the Utility

The VISION:Results Quick Start Utility is a batch program you run in batch mode by submitting the provided JCL member:

Sample Execution JCL — INFORM.JCL(RESULTQS)

Note: You can also run this utility online under the Definition Processor Main Menu
Import option.

Refer to the *Advantage VISION:Inform Definition Processor Reference Guide*.

This JCL contains an instream procedure, followed by the step to execute the procedure.

Specifying the Execution JCL

Figure 17-2 show the sample JCL contained in INFORM.JCL in the member RESULTQS. Review this JCL carefully.

At a minimum, make the following changes before submitting the JCL:

1. Supply a JOB card.
2. Supply procedure variables values—for information, see Specifying Procedure Variables.
3. Make modifications, as needed, to conform to your installation's standards.

The following figure shows the JCL to run the VISION:Results Quick Start Utility.

```
//* MEMBER RESULTQS                                                     00010000
//********************************************************************* 00020000
//* EXECUTE THE RESULTS QUICK START UTILITY TO CONVERT VISION:RESULTS * 00030000
//* OR VISION:EIGHTY DEFINITIONS INTO VISION:INFORM FORMAT.           * 00040000
//*                                                                   * 00050000
//* ***** NOTE *****                                                  * 00060000
//* THE SYSCOPY DD STATEMENT IS USED FOR MVS COPYBOOK LIBRARIES.      * 00070000
//* THE PANDD1  DD STATEMENT IS USED FOR PANVALET COPYBOOK LIBRARIES. * 00080000
//* THE MASTER  DD STATEMENT IS USED FOR LIBRARIAN COPYBOOK LIBRARIES.* 00090000
//*                                                                   * 00100000
//* THIS UTILITY MAY ALSO BE INVOKED INTERACTIVELY UNDER TSO/ISPF     * 00110000
//* USING THE DEFINITION PROCESSOR COMPONENT IMPORT FUNCTION.         * 00120000
//********************************************************************* 00130000
//RESLTQS PROC RGN=2M,                                                  00140000
//             LOADLIB=,                                                00150000
//             DEFLIB=,                                                 00160000
//             MEMBER=,                                                 00170000
//             RSLTLIB=,                                                00180000
//             RSLTDEF=                                                 00190000
//CONVRT EXEC PGM=RESULTQS,REGION=&RGN                                  00200000
//STEPLIB  DD DISP=SHR,DSN=&LOADLIB                                     00210000
//SYSPRINT DD SYSOUT=*                                                  00220000
//*SYSCOPY DD DISP=SHR,DSN=USER.RESULTS.COPYLIB                         00230000
//*PANDD1  DD DISP=SHR,DSN=USER.PANVALET.LIBRARY                        00240000
//*MASTER  DD DISP=SHR,DSN=USER.LIBR.MASTER                             00250000
//SYS004   DD DISP=OLD,DSN=&DEFLIB(&MEMBER)                             00260000
//SYSIN    DD DISP=SHR,DSN=&RSLTLIB(&RSLTDEF)                           00270000
//       PEND                                                           00280000
//********************************************************************* 00290000
//*  FOLLOWING IS A SAMPLE EXECUTION OF THIS PROCEDURE. BEFORE YOU    * 00300000
//*  RUN THIS PROCEDURE, SPECIFY:                                     * 00310000
//*                                                                   * 00320000
//*    RGN     - THE REGION SIZE. DEFAULT IS 2M.                      * 00330000
//*    LOADLIB - THE INFORM INSTALLATION LOAD LIBRARY.                * 00340000
//*    DEFLIB  - THE LIBRARY(PDS) TO CONTAIN THE INFORM DEFINITIONS.  * 00350000
//*    MEMBER  - THE PDS MEMBER NAME FOR THE CONVERTED VISION:INFORM  * 00360000
//*              FILE DEFINITION IN THE DEFINITION LIBRARY.           * 00370000
//*    RSLTLIB - THE PDS CONTAINING THE VISION:RESULTS FILE           * 00380000
//*              DEFINITION SOURCE STATEMENTS.                        * 00390000
//*    RSLTDEF - THE PDS MEMBER NAME OF THE INPUT VISION:RESULTS      * 00400000
//*              FILE DEFINITION TO BE CONVERTED.                     * 00410000
//*                                                                   * 00420000
//* *** N O T E ***                                                   * 00430000
//*                                                                   * 00440000
//*  THIS PROCEDURE ASSUMES INPUT FROM A PDS MEMBER. OPTIONALLY, IT   * 00450000
//*  MAY ALSO COME FROM A RESULTS COPY (MVS PDS), COPYP (PANVALET),   * 00460000
//*  OR COPYL (LIBRARIAN) STATEMENT. IF SO, YOU MUST UN-COMMENT THE   * 00470000
//*  APPROPRIATE SYSCOPY (MVS PDS), PANDD1 (PANVALET), OR MASTER      * 00480000
//*  (LIBRARIAN) DD STATEMENT IN THE PROCEDURE, SPECIFYING THE        * 00490000
//*  PROPER DATA SET NAME FOR THE LIBRARY USED. PLEASE REFER TO THE   * 00500000
//*  MANUAL FOR DETAILS IN SETTING UP COPY SUPPORT.                   * 00510000
//********************************************************************* 00520000
//STEP01 EXEC RESLTQS,RGN=2M,                                           00530000
//            LOADLIB='INFORM.LOADLIB',                                 00540000
//            DEFLIB='INFORM.DEFLIB',                                   00550000
//            MEMBER=FILENAME,                                          00560000
//            RSLTLIB='RESULTS.FILEDEFS',                               00570000
//            RSLTDEF=FILENAME                                          00580000
```

Figure 17-2 Sample Execution JCL for the VISION:Results Quick Start Utility, RESULTQS

Specifying Procedure Variables

The following summarizes the variables you need to specify when executing the supplied VISION:Results Quick Start Utility JCL procedure, RESULTQS.

Variable Name	Description
RGN	Specify the region size for the utility execution. The default is 2 MB.
LOADLIB	Specify the VISION:Inform installation load library. The default is INFORM.LOADLIB.
DEFLIB	Specify the data set name of a PDS into which the converted file definition will be placed. The default is the VISION:Inform Definition Processor definition library PDS.
MEMBER	This is the output PDS member name that the converted definition will use. Match the member name to the file name of the file definition being converted.
RSLTLIB	Specify the data set name of a PDS containing the VISION:Results file definition to be converted.
RSLTDEF	Specify the PDS member name of the VISION:Results file definition to be converted.

Figure 17-3 JCL Procedure Variables for VISION:Results Quick Start Utility

Providing DD Statement Overrides

Use DD overrides to change the specifications for the SYSIN data set from the default of a PDS member. You can do this in one of several ways:

Specifying a Sequential Data Set

To specify a sequential data set as SYSIN input, override the SYSIN after initiating the procedure with the following JCL DD override:

```
//CONVRT.SYSIN DD DISP=SHR,DSN=USER.SEQ.FIELDEF
```

Specifying Instream Data

To specify the SYSIN item to be converted as instream data, override SYSIN after the initiating the procedure as follows:

```
//CONVRT.SYSIN    DD *
        (input to be converted)
```

Specifying an MVS Copybook

To specify input using MVS copybook support:

1. Uncomment the SYSCOPY DD statement in the procedure.
2. Specify the data set name of a PDS containing the VISION:Results file definition to be copied.
3. Override the SYSIN DD statement with instream data as follows:

```
//CONVRT.SYSIN    DD *
  COPY membername
```

Specifying a CA-Panvalet Library Copybook

To specify input using CA-Panvalet library copybook support:

1. Uncomment the PANDD1 DD statement in the procedure.
2. Specify the data set name of the CA-Panvalet library containing the VISION:Results file definition to be copied.
3. Then override the SYSIN DD statement with instream data as follows:

```
//CONVRT.SYSIN    DD *
  COPYP membername
```

Specifying a CA-Librarian Library Copybook

To specify input using CA-Librarian copybook support:

1. Uncomment the MASTER DD statement in the procedure.
2. Specify the data set name of the CA-Librarian master file containing the VISION:Results file definition to be copied.
3. Then override the SYSIN DD statement with instream data as follows:

```
//CONVRT.SYSIN    DD *
  COPYL membername
```

Operational Characteristics

The VISION:Results Quick Start Utility's operational characteristics include:

- Supported Statement Types.
- Converted VISION:Inform File Definition.
- Member Naming Conventions.

Supported Statement Types

The VISION:Results Quick Start Utility processes all valid VISION:Results FILE and FIELD definition freeform statements, as well as the COPY, COPYP, and COPYL statements.

- Other valid VISION:Results processing and reporting statements can be included in the input, but will be ignored unless they contain errors.
- Make the VISION:Results FILE statement the first statement in the definition.

Converted VISION:Inform File Definition

The VISION:Results Quick Start Utility generates VISION:Inform file definition statements of the following types:

- FD (File Definition)
- LS (Segment Definition)
- L0 (Field Definition)
- Ln (Field Column Heading Definition)
- LX (Alternate Name Definition)
- Dn (Field Description Definition)

Converted Characters

The utility scans all text fields in the generated L0, Ln, LX, and Dn statements for the single quote ('), double quote ("), comma (,) and VISION:Inform system delimiter (from M4PARAMS) characters. If the utility encounters these characters, the utility changes them to either an underscore (_) or an “at” sign (@).

Generated FD Statement

The utility builds the generated FD statement from the VISION:Results FILE statement.
It uses the same file name as the VISION:Results file name and uses the name as the MEMBER JCL PROC parameter of the PDS member name of the converted definition.

Generated Key

If you specify a key length and location in the VISION:Results definition, the utility saves the values and uses them to assign a key field to the converted FD.

If you do not specify a key location in the VISION:Results FILE statement, the utility arbitrarily designates the first field output in the converted FD as the key and issues a diagnostic warning message so that you can inspect the converted FD and change it, if necessary.

Converted Record Format

The utility converts the record format, where possible.

If the utility could not translate the record format from the VISION:Results definition, it inserts a question mark (?) in the converted record format field and issues a diagnostic warning message so that you can review this field in the converted FD.

Generated Record and Block Size

For record and block size, the utility specifies the VISION:Results characters per record and characters per block fields as the VISION:Inform record size and buffer size fields, respectively, unless the record format is FIXED or ISAM, in which case, the utility sets the record size and calculates the records per block for the VISION:Inform FD statement.

Generated LS Statement

The utility generates the LS statement with a name of SEGMENT1, since VISION:Results file definitions represent flat, not hierarchical, files. It is also assigns a segment number of 1 and a level number of 1.

Generated L0 Statements

The utility always generates L0 statements with a generated field name of a 1-byte prefix (the character F), followed by a 7-digit number that is incremented by one for each field defined. The utility uses the VISION:Results field name, which can be up to 50 characters in length, to generate an alternate name for the field. For more information, see Generated LX Statements.

If the name is greater in length than the 30 bytes allowed for VISION:Inform alternate names, the utility uses only first 30 characters and then uses the entire 50-byte name as the field description. For more information, see Generated Dn Statements. When this happens, check the field definition for duplicate names using the Duplicate Name Check function of the new Definition Processor IMPORT panel.

The utility directly translates field location, length, type, output edit length, and rounding information to the L0 statement from the VISION:Results definition. It translates edit codes as closely as possible.

The utility generates a field location of four question marks (????) if the VISION:Results definition uses 5 bytes without a leading zero to define the field location.

Generated Ln Statements

The utility generates Ln statements for column headings from the VISION:Results field definition column heading information, if present. If you do not specify a column heading in the input VISION:Results definition, then the utility uses the field name as the column heading. Since you can specify column headings to be specified in 16-byte lines in VISION:Inform, check the converted definition in case the utility broke up the 30-byte VISION:Results column heading inappropriately during the conversion.

Generated Dn Statements

The utility generates Dn statements (field descriptions) based on the VISION:Results field name if it was greater than 30 bytes. If not, the utility uses the column heading as the field description.

Generated LX Statements

The utility generates LX statements for alternate names based on the first 30 characters in the VISION:Results field name. If there is more than one field that is greater than 30 bytes in the VISION:Results definition, the utility could generate duplicate names in the converted VISION:Inform FD. Use the Definition Processor duplicate check function on the converted FD to resolve any duplicate names.

Member Naming Conventions

The utility always uses the member name of the converted VISION:Inform definition from the RESULTQS JCL MEMBER PROC parameter. If this name exists in the output PDS, the utility overwrites it. For ease of use with the VISION:Inform Definition Processor, always make the PDS name match the file name of the file definition being converted.

Generated Output

The VISION:Results Quick Start Utility generates a SYSPRINT listing containing the following information:

- A listing of the VISION:Results input statements.
- Error messages for any input statements that are invalid.
- Informational or warning messages about the converted FD statements.

COPYP and COPYL Support

Before you use the COPYP (CA-Panvalet copy) and COPYL (CA-Librarian copy) statements, you need to link the appropriate interface routines with the VISION:Results Quick Start Utility for CA-Panvalet or CA-Librarian access prior to running the VISION:Results Quick Start Utility.

For more information, see the *Advantage VISION:Inform Installation Guide for CICS*.

Index