Advantage™ VISION:Inquiry® for CICS®

Technical Reference Guide

6.5

Computer Associates®

IQRFC065.PDF/D07-402-010

Contents

Chapter 1: Introducing VISION:Inquiry

Chapter 2: System Overview

Chapter 3: System Components

Chapter 4: The Definition Process

Chapter 5: The Utilities

Chapter 6: Programming and Operation Considerations

Chapter 7: User Exits

Appendix A: System Vocabulary and Codes

Appendix B: Sample User Exits

Appendix C: The IXSECTY and IXSGEN Macros

Appendix D: System Modules

Appendix E: VISION:Inquiry Target and Distribution Libraries

Index

1 Introducing VISION:Inquiry

Thank you for choosing Advantage™ VISION:Inquiry® 6.5 (hereinafter referred to as VISION:Inquiry). Before you install the software, read this chapter for important information.

CD-ROM Contents

- Online documentation
- Adobe® Acrobat® Reader software and Acrobat Help

About the Online Documentation

The CD-ROM contains the documentation for VISION:Inquiry. The documents, called books, are in Adobe Acrobat Portable Document Format (PDF) and are designed for you to read online using the Acrobat Reader.

Each online document contains a table of contents, index, and cross-references.

Note: You can install the online documentation only on a Windows® system.

Installing Online Documentation and the Acrobat Reader

You can install the online documentation on your local hard drive or on a network server. Alternately, you can access the documentation directly from the CD-ROM.

If you do not have Acrobat Reader installed, you can install it from the CD-ROM.

To install the online documentation, the Acrobat Reader, or both:

1. Close all application programs.
2. Insert the CD-ROM into the CD-ROM drive.
3. Click the Start menu and select Run.
4. In the Run dialog box, type: D:\Books\Setup.exe (where D:\ is the CD-ROM drive) and click OK.

5. Follow the instructions. Computer Associates recommends that you install the online documentation in the default directory (C:\ProgramFiles\CA\Advantage VISION_Inquiry 6.5 CICS\Books\) or a directory of your choice (for example, C:\Advantage VISION_Inquiry 6.5 CICS\Books\).

Viewing Online Documentation

Regardless of the location of the online documentation (on a local drive, a network server, or CD-ROM), you can view the online documentation using the following methods:

- In Windows, click the Start menu, point to Programs, point to Advantage VISION_Inquiry 6.5 CICS. Double-click the PDF file name.
- In Windows Explorer, point to the Books directory on the hard drive where you installed the online documentation. Double-click the PDF file name.
- In Windows Explorer, point to the Books directory on the CD-ROM drive and double-click the PDF file name.

Using Adobe Acrobat Reader

Use Acrobat Reader to view the online documentation, adjust the size of the page, and perform searches. For more information, use the Acrobat Help menu.

Contacting Total License Care (TLC)

TLC is available Monday-Friday 7 am - 9 pm Eastern Time in North America and 7 am - 7 pm United Kingdom time. Additionally, 24-hour callback service is available for after hours support. Contact TLC for all your licensing requirements.

Be prepared to provide your site ID for product activation.

To activate your product, use one of the following:

North America:	800-338-6720 (toll free) 631-342-5069	help@licensedesk.cai.com
Europe:	00800-1050-1050	euro.tlc@ca.com

If your company or local phone service does not provide international access, please call your local Computer Associates office and have them route you to the above number.

Australia:	1-800-224-852
New Zealand:	0-800-224-852
Asia Pacific:	800-224-852
Brazil:	55-11-5503-6100
Japan:	JPNTLC@ca.com

Contacting Computer Associates

For technical assistance with this product, contact Computer Associates Technical Support on the Internet at SupportConnect.ca.com. Technical support is available 24 hours a day, 7 days a week.

Product Description

VISION:Inquiry is an easy-to-use general purpose, multilingual application program for inquiry and retrieval of data from IMS™ (DL/I) databases, DB2® tables or views, and VSAM data sets. Data is retrieved in response to inquiries consisting of simple user-oriented commands.

VISION:Inquiry features include:

- Immediate online response to inquiries.
- An easy-to-learn free-form natural inquiry language.
- Immediate online response to inquiries using the Automatic Query Facility (AQF), a complete menu-driven system with fill-in-the-blank specifications.
- Modifiable inquiry language vocabulary.
- Access to multiple IMS databases, DB2 tables/views, and VSAM data sets, or a combination of the three.
- Access to DB2 tables using embedded SQL statements.
- Optional User Defined Output (UDO) formatting.
- Selective terminal output routing.
- Arithmetic commands and functions.
- Logical commands.
- Report summary commands.

- An interactive capability for creating and storing inquiries.
- Complete text editing capabilities for stored inquiries.
- Conversational mode or continuous mode of operation.
- Built-in data security.
- User exit facilities.
- Vocabulary and messages translatable to languages other than English.
- Transfer of data in different formats from host to PC, if VISION:Journey® for Windows® is installed.
- Advantage™ Intraccess™ (hereafter referred to as Intraccess) support. Intraccess is a Java-based tool that communicates with VISION:Inquiry using TCP/IP (Transmission Control Protocol/Internet Protocol). Use it to run queries stored in the VISION:Inquiry system data base, deliver the data to PCs, and make the data available to end users. For more information about the Intraccess product, see the Intraccess documentation.

DB2 Information

This guide contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing DB2 is specifically applicable to DB2 licensed sites.

Intraccess Information

This guide contains information for sites licensed with the Intraccess option. Text containing Intraccess is specifically applicable to Intraccess licensed sites. See the Intraccess documentation for more information about the Intraccess option.

Objectives of this Guide

This guide is written for database administrators and other data processing personnel responsible for supporting VISION:Inquiry. It provides information for installing and implementing VISION:Inquiry at your facility. It also presents information about some of the more complex capabilities of VISION:Inquiry. With this information you may assist your end users to make the most of the VISION:Inquiry retrieval and reporting capabilities.

The guide presents complete information for interfacing VISION:Inquiry to your specific software environment as well as to your user databases, tables, and data sets.

Note: Throughout this document, OS/390® is synonymous with z/OS™, unless specified differently.

How this Guide Is Organized

- Chapter 1, "Introducing VISION:Inquiry" provides a brief introduction to VISION:Inquiry and to this guide.
- Chapter 2, "System Overview" gives you a system overview of VISION:Inquiry.
- Chapter 3, "System Components" explains the system components.
- Chapter 4, "The Definition Process" explains how to define your databases, tables and views, data sets, logical terminals, and directories to VISION:Inquiry.
- Chapter 5, "The Utilities" explains the utilities.
- Chapter 6, "Programming and Operation Considerations" explains programming and operations considerations.
- Chapter 7, "User Exits" provides you with guidelines for coding and implementing VISION:Inquiry user exits.
- Appendix A, "System Vocabulary and Codes" contains system vocabulary and codes.
- Appendix B, "Sample User Exits" contains sample user exits.
- Appendix C, "The IXSECTY and IXSGEN Macros" describes IXSECTY and IXSGEN macros.
- Appendix D, "System Modules" describes the system modules.
- Appendix E, "VISION:Inquiry Target and Distribution Libraries" contains information about the content of the VISION:Inquiry target and distribution libraries.

VISION:Inquiry Configuration Requirements

- Operates in a CICS® environment or as a batch program.
- Uses the paging and routing facility of BMS and standard coding and design conventions for application programs executing in the CICS environment.
- Supports the display and printer terminals supported by CICS Basic Mapping Support (BMS).

Product Terminology

The following terms (in alphabetical order) are used to reference different parts of VISION:Inquiry.

Automatic Query Facility (AQF)	The menu driven portion of VISION:Inquiry. AQF uses different menus to build the inquiry and then passes control to native VISION:Inquiry to process the inquiry and send the output to the terminal.
Database database map user database	Database, database map, and user database apply equally to IMS (DL/I) databases, DB2 tables/views, and VSAM data sets except where specifically noted.
Intraccess	A Java-based tool that communicates with VISION:Inquiry using TCP/IP. Use it to run queries stored in the VISION:Inquiry system database, deliver the data to PCs, and make the data available to end users.
Front-end and main programs	The front-end program is the first program that gains control in the native VISION:Inquiry processing and builds the input area containing the inquiry. The front-end program passes the input area to the main program using the CICS LINK command to process and return the output.
Inquiry or query	A request for data from a database, based on specified conditions. The words inquiry and query are used interchangeably.
Native SQL Syntax	A facility of native VISION:Inquiry which provides you with the capability to use embedded SQL SELECT statements to access DB2 tables.

Native VISION:Inquiry	The free-form portion of VISION:Inquiry (not including AQF).
VISION:Journey for Windows	Facility which provides users with the capability to download the data or report to a PC file using Microsoft® Windows. VISION:Journey is licensed separately.
VISION:Journey download data set	Staging file used to hold data for download processing. For VISION:Journey, this data set is a VSAM KSDS data set. The person who installs the product must specify the data set type at installation time.
VISION:Inquiry	The complete system, comprising Native VISION:Inquiry and AQF.
VSAM hierarchical data sets	VSAM data sets with fixed or variable occurrences of data items. These types of VSAM data sets are defined differently for VISION:Inquiry than other VSAM data sets which, in this guide, are called *VSAM non-hierarchical data sets*.

The following notation is used in the VISION:Inquiry documents:

DB2 tables/views	DB2 tables or views.
DL/I database	Same as IMS (DL/I) database.
LLLLLL (MMMMMM)	Indicates that MMMMMM is a member of the LLLLLL library.

Associated Documents

The following books are available for VISION:Inquiry. All of the VISION:Inquiry books are on the VISION:Inquiry documentation compact disc.

Advantage VISION:Inquiry for CICS Getting Started	Contains a brief introduction to the product and an overview of the installation. This document was previously known as the CD booklet.
Advantage VISION:Inquiry for CICS Release Summary	This environment-specific document contains practical techniques for using VISION:Inquiry more efficiently. It also contains information pertinent to new releases of the system as well as useful information from our customers and Technical Support staff. ■ Use this environment-specific document in conjunction with its corresponding *Advantage VISION:Inquiry for CICS Technical Reference Guide.* ■ This document was previously called the *VISION:Inquiry Customer Bulletin*.
Advantage VISION:Inquiry for CICS Installation Guide	This environment-specific document contains the installation instructions and the Post-Installation Dialog.
Advantage VISION:Inquiry for CICS Technical Reference Guide	This environment-specific document contains descriptions of the system components, information on defining the system, using the system utilities, programming and operation considerations, user exits, system modules and system macros. ■ Use this environment-specific document, in conjunction with its corresponding *Advantage VISION:Inquiry for CICS Release Summary*. ■ This document was previously called the *VISION:Inquiry Technical Reference Manual*.

Advantage VISION:Inquiry for IMS and CICS Automatic Query Facility (AQF) User Guide	Provides end users with information about using VISION:Inquiry Automatic Query Facility (AQF). It explains the use of the menu-driven system and provides examples of simple and complex inquiries. This document may be referred to as the *AQF User Guide*. This document was previously called the *VISION:Inquiry for IMS and CICS Automatic Query Facility User's Guide*.
Advantage VISION:Inquiry Reference Guide	Provides end users with information about using VISION:Inquiry. It explains command statements, provides syntax rules, and contains examples showing how the language works. ■ This document was previously called the *VISION:Inquiry for IMS, CICS, and TSO User's Guide*. ■ The *VISION:Inquiry Reference Summary* is now included in an appendix in the *Advantage VISION:Inquiry Reference Guide*. It is an easy-to-use summary of VISION:Inquiry statements and commands and their relationships to each other.
Advantage VISION:Inquiry Messages Guide	Lists and explains the various input and output error messages that can be issued by AQF, native VISION:Inquiry, and VISION:Journey for Windows. This document was previously called *VISION:Inquiry for IMS, CICS, and TSO Error and Informational Messages*.

The following is a list of associated documents for VISION:Inquiry:

VISION:Journey for Windows System Administrator's Guide	Provides the hardware and software requirements, the PC software installation procedure, and the steps necessary by the system administrator to establish and maintain the user profiles for VISION:Journey for Windows. This document is on the VISION:Journey compact disc.
VISION:Journey for Windows User's Guide	Explains the menu commands, dialog boxes, and functions available to the end user and how to use them. This document is on the VISION:Journey compact disc.
Intraccess help	Explains the menu commands, dialog boxes, and functions available to both the user with administrative authority and the end user. This documentation is on the Intraccess compact disc.

2 System Overview

VISION:Inquiry is an effective tool for easy access to information stored in IMS (DL/I) databases, DB2 tables or views, and VSAM data sets. To use VISION:Inquiry, the end user needs relatively little data processing experience or knowledge of database and file structures, display terminal characteristics, or knowledge of the CICS environment.

This chapter contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing DB2 is specifically applicable to DB2 licensed sites. Terms such as database, database map, user database, and so on, refer equally to IMS (DL/I) databases, DB2 tables or views, and VSAM data sets, unless explicitly qualified.

VISION:Inquiry uses an easy-to-learn, natural inquiry language that allows the end user to select information from databases or data sets, manipulate it, perform computations, and format it into professional looking, easy-to-read displays. VISION:Inquiry also provides facilities for those who need or prefer to design their own output.

VISION:Inquiry automatically returns output to the terminal from which the inquiry originated, but can optionally direct the output to other terminals or printers.

VISION:Inquiry validates inquiry statement syntax interactively and sends back error messages to the terminal. Once an inquiry is composed, it can be stored or executed immediately; however, only syntactically correct inquiries can be stored or executed.

VISION:Inquiry Features and Capabilities

VISION:Inquiry capabilities include:

- Immediate online access to one or two IMS (DL/I) databases, DB2 tables/views, VSAM files, or a combination of them using AQF (Automatic Query Facility)
- Immediate online access to as many as 16 databases at once using native VISION:Inquiry
- Access to DB2 tables using embedded SQL statements
- Modifiable user language vocabularies
- Capability to convert the vocabulary to languages other than English
- User and data security by application and terminal ID or user ID
- Creation, storage, and recall of inquiries and arithmetic functions
- Modification of stored inquiries
- Operation in CICS and batch environments
- Checkpoint and non-checkpoint modes of operation
- Printer support
- Generic user exits for a variety of functions
- System utility programs for creating and maintaining the system database
- Documented error messages for components of the system
- Logical and physical hierarchical IMS (DL/I) database access
- Access to fixed and variable length segments/records
- Concatenated key support
- Secondary and alternate indices
- Virtual logical child access for IMS (DL/I) databases.

VISION:Inquiry Processing Capabilities

- Arithmetic computations on specified fields
- Parenthetical arithmetic expressions
- Temporary fields
- Grand and subtotal, count, and average of specified fields
- Data extracted from DL/I databases, DB2 tables/views, and VSAM files to CICS extra partition transient data sets or an OS sequential file in batch environment.

VISION:Inquiry Display Capabilities

- Display of data from specified fields
- Automatically formatted horizontal (columnar) or vertical (row) output
- Conditional selection of data
- Sort output data into a specified order
- User-defined output formats for the page or screen
- Direct output to other terminals or a printer
- Display sample data
- Display of current date, time, and page number
- Partial fielding.

VISION:Journey for Windows Capabilities

- Access VISION:Excel®, VISION:Inquiry, VISION:Inform®, and VISION:Results™ from a PC to generate simple or complex queries and reports
- Run queries against VISION:Inquiry host VSAM, DB2 and IMS (DL/I) databases
- Download results of queries into a variety of PC DOS file formats, such as ASCII, dBASE, 1-2-3, Symphony, and Excel
- Obtain information from the enterprise server easily, without having to know rigorous logon and logoff procedures
- Access the full functionality of enterprise server products for information management while benefiting from the user-friendly, easy-to-use PC graphical environment.

Intraccess Capabilities

- List and run queries stored in the system data base and either download the results into a PC file or send the results to a terminal or printer.

All of these functions are available to users at terminals and PC work stations accessed through VISION:Inquiry statements. The *Advantage VISION:Inquiry Reference Guide* and the *Advantage VISION:Inquiry for IMS and CICS Automatic Query Facility (AQF) User Guide* contain many examples of outputs that VISION:Inquiry can generate; they also provide detailed instructions for using the VISION:Inquiry statements.

VISION:Inquiry System Components

The native VISION:Inquiry system consists of a system database, the inquiry processing programs, and utility programs. This chapter briefly describes each of these components. Chapter 3, "System Components", explains how these components relate to each other and to the rest of the system.

The Automatic Query Facility (AQF) consists of a set of screens (BMS maps) and programs. This guide describes the function and relation of these components.

The VISION:Inquiry system database is the heart of the system. The utility programs are used to create and maintain the system database.

The VISION:Journey host component consists of a VSAM KSDS download data set, file transfer processing programs, and utility programs.

Additionally, VISION:Inquiry has a front-end program to access the Intraccess option.

The VISION:Inquiry System Database

The system database is a root-only DL/I (HDAM) database, a DB2 table, or a VSAM (RRDS) file. The elements within the database can contain variable length data that can span multiple segments/records. However, the VISION:Inquiry access methods used only recognize fixed length segments or records.

The elements within the database are used to interpret user inquiries and to provide relevant information to the VISION:Inquiry programs. The system database consists of the following:

- High level index
- Item index of all entries in the database
- User vocabulary or vocabularies
- User database maps
- User directories including stored inquiries
- All diagnostic error messages
- Terminal description
- Conversational scratch pad storage.

The Inquiry Processing Programs

The inquiry processing programs enable the end user to create, store, modify, and execute inquiries. The processing programs check the inquiries for syntax errors and consistency with the information stored on the system database; messages are issued when errors are detected.

The user can execute or store error-free inquiries in the system database.

When an inquiry is executed, its requests for database information are converted into DL/I calls, SQL calls, or VSAM I/O requests, which retrieve the requested data from the specified database. The program also performs any additional selection, sorting, arithmetic processing, and formatting specified in the inquiry.

The syntax of inquiries is the same for all purposes. Special considerations and restrictions for each environment are discussed in Chapter 6, "Programming and Operation Considerations".

The CICS Program

The VISION:Inquiry for CICS inquiry program executes as a transaction in the CICS region.

The inquiry program returns error messages to the originating terminal. Normally, the inquiry program sends output from the inquiries to the originating terminal using the paging facility of BMS. Optionally, you can direct the output to another terminal or a printer using the routing facility of BMS.

The Batch Program

The VISION:Inquiry batch inquiry program executes as a batch job. Input to the batch program is in 80-byte logical records.

Once the inquiry is processed, the batch inquiry program directs the output to a system output data set (SYSOUT) for subsequent printing.

You can also run the batch program as BMP (Batch Message Processing) program using DBCTL facility. BMPs perform batch type processing online and can access databases controlled by DBCTL.

VISION:Inquiry System Flow

This section illustrates the interaction of VISION:Inquiry with other system components, such as database managers and transaction monitors. Each section depicts the interactions within one environment for a typical system configuration.

- The illustrations are accompanied by explanations of what happens when a user enters an inquiry at a remote terminal.
- The circled index numbers in the illustrations correspond to the explanations.

VISION:Inquiry Online Processing

Figure 2-1 illustrates the interactions of native VISION:Inquiry when it executes in the CICS region.

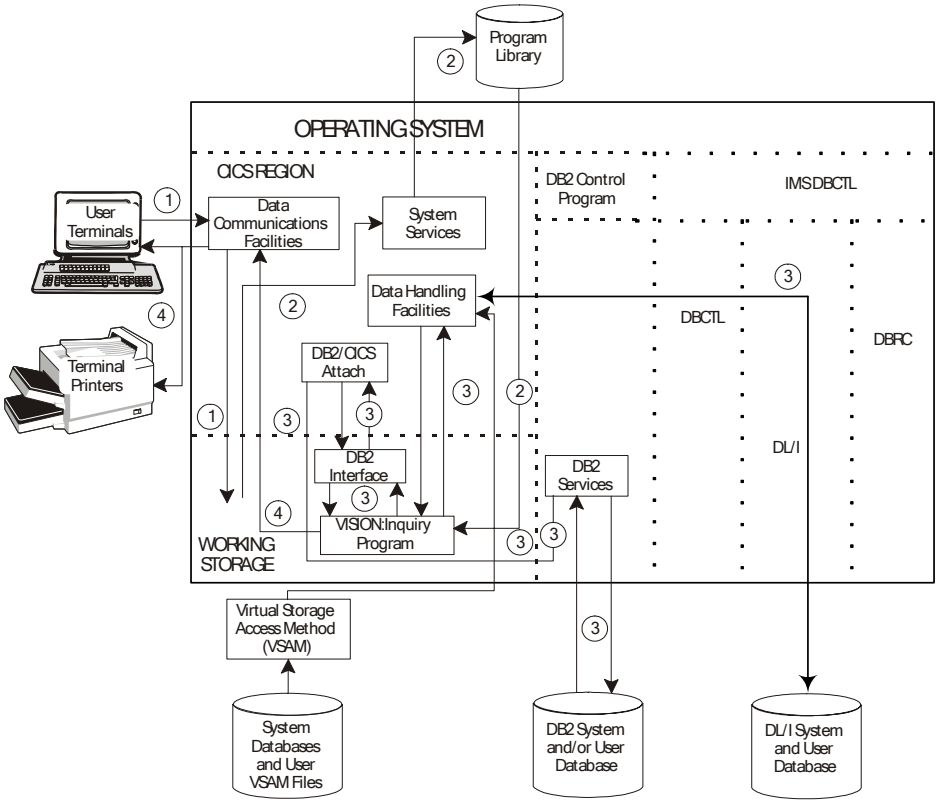


Figure 2-1 The Interactions Between VISION:Inquiry and CICS

The following describes the interactions shown in Figure 2-1.

1. Each inquiry passes from the originating terminal to the CICS control region for scheduling.
2. The system services facility interprets the transaction identifier and loads the VISION:Inquiry program from the program library.
3. VISION:Inquiry retrieves the inquiry as an input message, checks the syntax, then issues CALL/READ commands to the appropriate databases or VSAM files to retrieve the data. The database calls are processed by DL/I in IMS DBCTL region or DB2 CICS Attach, and VSAM reads are processed by the CICS File Management Program. Note that the system database can be a DL/I database, a DB2 table, or a VSAM file.
4. The processing program returns the output from the processed inquiry to the appropriate terminal, through a message queue.
5. Figure 2-1 shows DL/I as part of the CICS control region (local DL/I), but it does not need to run there. You can run DL/I in its own address space called DBCTL facility.

VISION:Inquiry Batch Processing

Figure 2-2 illustrates the interactions of VISION:Inquiry when it executes as a batch program. For illustrative purposes, the CICS region is omitted from Figure 2-2.

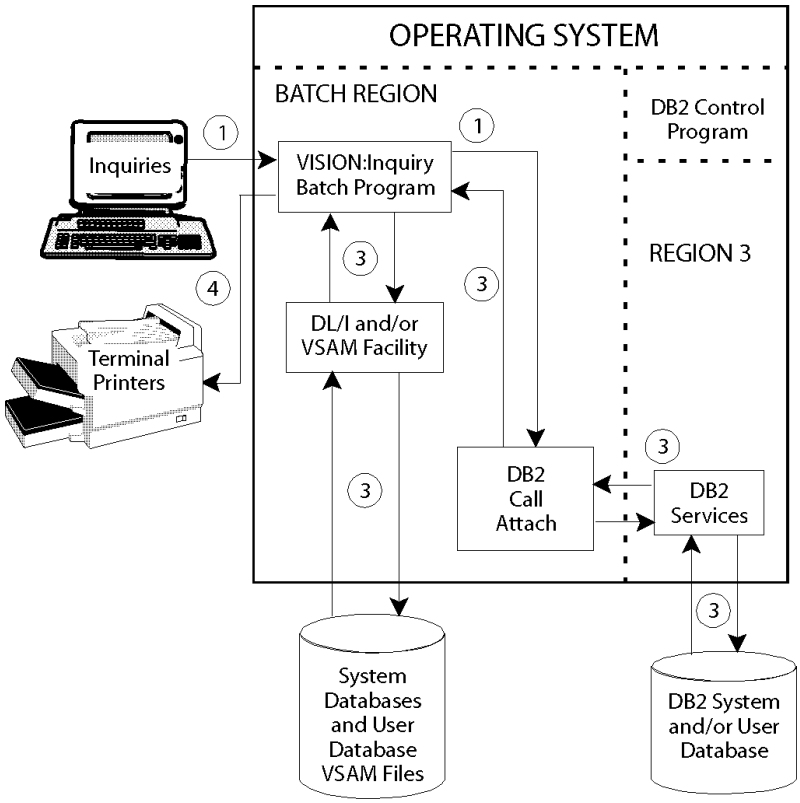


Figure 2-2 The Interaction with VISION:Inquiry Operating in the Batch Region

The following describes the interactions illustrated in Figure 2-2.

1. You submit a batch job to start VISION:Inquiry in the batch region. The job allocates all the VISION:Inquiry files and passes the appropriate parameters to VISION:Inquiry.
2. You supply the inquiries as 80-byte records (usually submitted with the JCL for invoking VISION:Inquiry). You can submit one or more inquiries and each can specify the same or a different transaction identifier.
3. VISION:Inquiry retrieves inquiries from an input data set, checks the syntax, and issues calls to the appropriate databases or VSAM files to retrieve data; VSAM data sets are accessed using standard VSAM I/O. Note that the system database can be a DL/I database, a DB2 table, or a VSAM file.

4. VISION:Inquiry writes the output from the processed inquiries to the printer. VISION:Inquiry terminates when all the inquiries in the input stream are processed.

You can also run the Batch program as a BMP (batch message processing) program using DBCTL facility. BMPs perform batch type processing online and can access databases controlled by DBCTL.

VISION:Inquiry Text Editor Processing

The Text Editor facility is the stored inquiry editor component of the VISION:Inquiry system. It provides full editing capability for stored inquiries. Through the Text Editor, users can display stored inquiries, edit them, and then save or execute the edited inquiries.

The Text Editor uses the CICS Temporary Storage facility to keep the intermediate results.

VISION:Inquiry AQF Processing

AQF, Automatic Query Facility, is the menu-driven component of the VISION:Inquiry system. AQF automatically creates and executes inquiries using a fill-in-the-blanks method. It is ideal for the first time or occasional user as it requires no knowledge of VISION:Inquiry syntax. The user need not be familiar with the VISION:Inquiry databases, field names, or commands.

AQF is a simple, easy way for a person with little or no data processing background to produce a variety of reports using information stored in IMS (DL/I) databases, DB2 tables, and VSAM data sets. By selecting options from the menu screens, you can access data fields in the databases and produce reports.

AQF executes in the CICS online region and uses the CICS Temporary Storage facility to keep the information about user selections on each screen.

VISION:Journey Processing

VISION:Journey allows the user to do all interactions on the PC. When information is required, the PC accesses the server, or host software (such as VISION:Inquiry). The communication between the workstation client and the enterprise server, VISION:Inquiry, is achieved through CICS or IMS. In other words, VISION:Journey, on the PC, logs on to CICS or IMS. It uses CICS or IMS transactions to extract and download data. VISION:Journey then logs off the enterprise server.

The extraction of requested data or reports is initiated by the VISION:Inquiry command PCE or OUTPUT 'dummy terminal'. VISION:Inquiry performs the extract, writes the extracted data or report and the other VISION:Journey for Windows control information to the download data set, and starts the VISION:Journey host computer component using the CICS START command. VISION:Journey then communicates with companion software resident in the PC workstation to download the output data or report.

The enterprise server component of VISION:Journey will be discussed in greater detail in this guide. For information regarding the PC component of VISION:Journey, see the VISION:Journey for Windows System Administrator's Guide.

VISION:Journey uses a KSDS VSAM download data set. For each extraction, the extracted data or report and the necessary information will be written to this download data set by VISION:Inquiry before download. The extracted data or report is assigned a unique subsequence number for each user. Using the unique subsequence number, download starts after all the extracted data for the inquiry is written to the download data set.

Intraccess Processing

Intraccess is a Java-based tool that communicates with VISION:Inquiry using TCP/IP. Users run queries stored in the VISION:Inquiry system database, deliver the data to PCs, and make the data available to end users. The user can get the list of the stored queries on the PC and then process the selected query. Based on the stored query content, for example, non-UDO vs. UDO (User Defined Output), the output data vs. report is delivered to the user's PC. The user also has the option of sending the output data/report to a CICS terminal or printer. A VISION:Inquiry front-end routine sets up the TCP/IP environment for handling stored queries submitted by Intraccess. It also uses the temporary storage facility of CICS for storing the output result before sending it to the user PC. See the Intraccess documentation for information about the Intraccess components.

VISION:Inquiry Native SQL Syntax Support

You can use embedded SQL SELECT statements in inquiries in native mode to access DB2 tables. You can display the output at the originating terminal or direct it to another terminal or printer. You can also store SQL syntax inquiries for later editing and execution.

User Exits

This section briefly describes logical exit points for interfacing with external user developed routines. These exit routines are used for special considerations specific to each installation.

The design, coding, and implementation of the exits must conform to specific conventions. The person coding the exits assumes the responsibility for maintaining the system's integrity.

Input exit	Use to modify inquiry statements. This exit is commonly used to add conditional selection clauses to inquiries. These clauses are added for the purpose of limiting access to specific data from certain logical terminals (that is, for value level security).
Output exit	Use to modify or delete the output fields of an inquiry.
Conversion exit	Use to perform data conversion of specified fields. Each time the specified field is referenced in an inquiry, the exit routine receives control and performs the desired conversion.
Function exit	Use to process multiple object fields and literals in order to produce a result. When the exit receives control, the fields are passed to it; they are processed, and the result is returned in a result field.
Security exit	Use to override the normal security of the system to allow security based on other than standard factors.
VSAM exit	Use to expand or change the VSAM records read from the user file before processing them.
EXTRACT exit	Use to override the default destination ID (DYL1) used by the EXTRACT command.
Sort exit	Use to delete the records from the VSAM sort work data set after the online sort completion.
CICS output handling exit	Use to handle the output data in a way other than the standard output processing of VISION:Inquiry which uses the BMS paging facility of CICS.

The VISION:Inquiry Utilities

Note: See Chapter 5, "The Utilities", for detailed descriptions.

This section briefly describes the utilities used for maintaining the VISION:Inquiry system. Some of these capabilities are:

- Create and maintain the system database
- Define and modify the user database and VSAM file description
- Define and modify the VISION:Inquiry system vocabulary
- Define and modify the VISION:Inquiry diagnostic error messages
- Provide statistical information reflecting the contents of the system database
- Modify the VISION:Inquiry executable code
- Reorganize and backup the system database
- Convert or unload stored inquiries
- Create and maintain the VISION:Journey download database

IIINIT Utility

The IIINIT utility initializes the system database by writing one or more space management records (rows) and formatting empty records for the index and directory.

IIINIT executes in batch. Space for the system database is allocated through JCL or IDCAMS. The index and directory sizes are specified in the input.

IIGEN Utility

The IIGEN utility is the most powerful of all the VISION:Inquiry utilities because it has multiple functions.

Based upon the user supplied input statements, IIGEN can do the following:

- Define VISION:Inquiry transaction codes
- Create or recreate error messages
- Create or recreate user vocabularies
- Create or recreate database maps
- Create, update, or recreate directories
- Create, update, or recreate logical terminal descriptions.

The IIGEN utility executes in the batch environment and is invoked through JCL statements. IIGEN performs its various functions through input statements. As a result of executing IIGEN, an output listing is produced which reflects its successful completion and lists messages for error conditions.

IXUUNLD Utility

The IXUUNLD utility unloads the system database. IXUUNLD copies each element of the system database to a variable length sequential file. This utility is used in conjunction with the IXULOAD utility for maintenance and backup of the system database.

IXUUNLD executes in the batch environment and is invoked through JCL statements. No input statements or parameters are required by the program.

The utility produces an output listing that reflects its successful completion and lists error messages for error conditions.

IXULOAD Utility

The IXULOAD utility reloads an unloaded system database to an initialized system database. The utility executes in the batch environment and is invoked through JCL statements.

At least one input statement (END) is required. The other input statements such as OPTIONS, INCLUDE, and EXCLUDE are optionally specified to selectively copy or not copy inquiries and functions. The included or excluded elements can be specified in any number of combinations.

The utility produces an output listing that reflects its successful completion and lists error messages for error conditions.

IXUSQRY Utility

The IXUSQRY utility unloads stored inquiries and functions. It copies the source of the stored inquiries and functions from the system database to an 80-byte sequential data set. This data set can then be used as input to the batch version of the product to restore the inquiries and functions. The IXUSQRY utility is primarily used during the recreation of directories which causes the stored inquiries and functions to be deleted.

IXUSQRY executes in the batch environment and is invoked through JCL statements. At least one input statement and the END statement is required.

The utility produces an output listing that reflects its successful completion and lists error messages for error conditions.

IXUIQRY Utility

The IXUIQRY utility converts the inquiries and functions stored from releases prior to the 6.0 release. The purpose of this utility is to convert the stored inquiries from internal to source format (which can then be used by the Text Editor).

The utility creates a sequential file containing the source inquiries and functions with the necessary control statements. This file can then be used as input to the batch version of the product to restore the inquiries and functions back in the new release.

IXUSTAT Utility

The IXUSTAT utility provides information about the components and their space utilization on the system database. This information is used to optimize the organization of applications, maps, and directories to minimize the number of system database calls made by the VISION:Inquiry processing programs. By running IXUSTAT periodically, you can also monitor the amount of space being used by stored inquiries, stored functions, and deferred inquiries.

IXUSTAT executes in the batch environment and is invoked through JCL statements. There are no input parameters or control statements for IXUSTAT.

IFUCLEN Utility

The IFUCLEN utility is the cleanup utility. It deletes the VISION:Journey extracted data or report written in the download data set. The utility executes in the batch environment and is invoked through JCL statements.

The input statements OPTIONS and END are specified to selectively delete the VISION:Journey extracted data or report in the download data set based on subsequence key, date of extraction, and so on.

The utility produces an output listing that reflects successful completion or lists error messages for error conditions and a second listing that gives a report of the records deleted.

3 System Components

The system database is a major component of the VISION:Inquiry system, containing information about users and user databases. This chapter describes the system database and its contents, as well as the VISION:Journey download database and its contents.

Test data, distributed with this product, are also described. The test data include the IMS (DL/I) test databases (PLANT and SKILL), the VSAM test data sets (VSSKILL, VSPLANT, VSHPLANT, and VSHSKILL), and the DB2 test tables/views. The terms “test data” and “test databases” can be used interchangeably.

This chapter contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing DB2 is specifically applicable to DB2 licensed sites.

The information in this chapter provides background information about the various system components. It is not required reading for the experienced database administrator or those familiar with VISION:Inquiry. However, this information can be the difference between a smooth installation of VISION:Inquiry and a difficult one. It can also help you to utilize your VISION:Inquiry system to its optimum efficiency.

Terms such as databases, database map, user database, and so on, refer equally to IMS (DL/I) databases, DB2 tables/views, and VSAM data sets unless explicitly qualified.

The System Database

The system database is implemented as an IMS (DL/I) HDAM database (OSAM or VSAM) consisting of root segments only, a two-column DB2 table, or a VSAM RRDS data set. Each segment holds 2040 bytes of data. (In this publication, the terms segment, record, and block as applied to the system database are used interchangeably to refer to a (root) segment of an IMS (DL/I) database, a row of a DB2 table, or a record of a VSAM RRDS data set.)

Within the system database, the following information is stored:

- Map of free and used system database records
- High level index
- Item index of all system database entries
- System vocabulary
- Text for most messages
- Maps of user databases, data sets, tables, and views
- User directories
- Terminal and printer descriptions
- Conversational scratch pad storage

The above items are variable in length and can occupy more than one block when stored in the database. This is common for database maps which, depending upon the number of segments, can be lengthy.

All entries in the database are chained to one another through forward and backward pointers. The bit map keeps track of where free space is located. Statistical information on the amount of free space is also maintained. All segments within the database are reflected in either the free space or used space statistics.

The system database can be shared among VISION:Inquiry tasks in different regions. Special considerations concerning contention problems are described in the installation procedures in the *Advantage VISION:Inquiry for CICS Installation Guide*.

The system database is one of the components of the system that greatly affects performance. It is constantly used for the translation of inquiries. If the database is excessively large, the result is slower translation. The number of users per system database may affect translation time and must be considered when planning your installation.

System Database Index

All items stored in the system database are located through the two levels of indices, *high level index* and *item index*. The indices are fixed in size and consist of a number of segments within the system database. Their size is specified when the initialization program IIINIT executes.

- Each index block contains 32 to 101 entries, depending on the type of entries. The first entry in the system database is always a high level index segment. All other high level and item index segments are chained to this through pointers. *Figure 3-1 on page 3-4* illustrates a typical index within the system database.
- Each **high level index** entry contains a 17-byte key and a pointer. The key is the highest key in each block of item index and the pointer points to that block. VISION:Inquiry uses these entries to locate a block of the item index.
- Each **item index** entry contains a 17-byte key. This key maintains the sequence by transaction code (TRANCODE), entry type (such as V for vocabulary, M for maps), and entry name (SYSVOCAB, MAP1, and so on). VISION:Inquiry uses these when it attempts to locate an element.

When VISION:Inquiry locates the desired key, a pointer within the entry points to the segment containing the described element. Other fields within the item index vary depending upon the type of element.

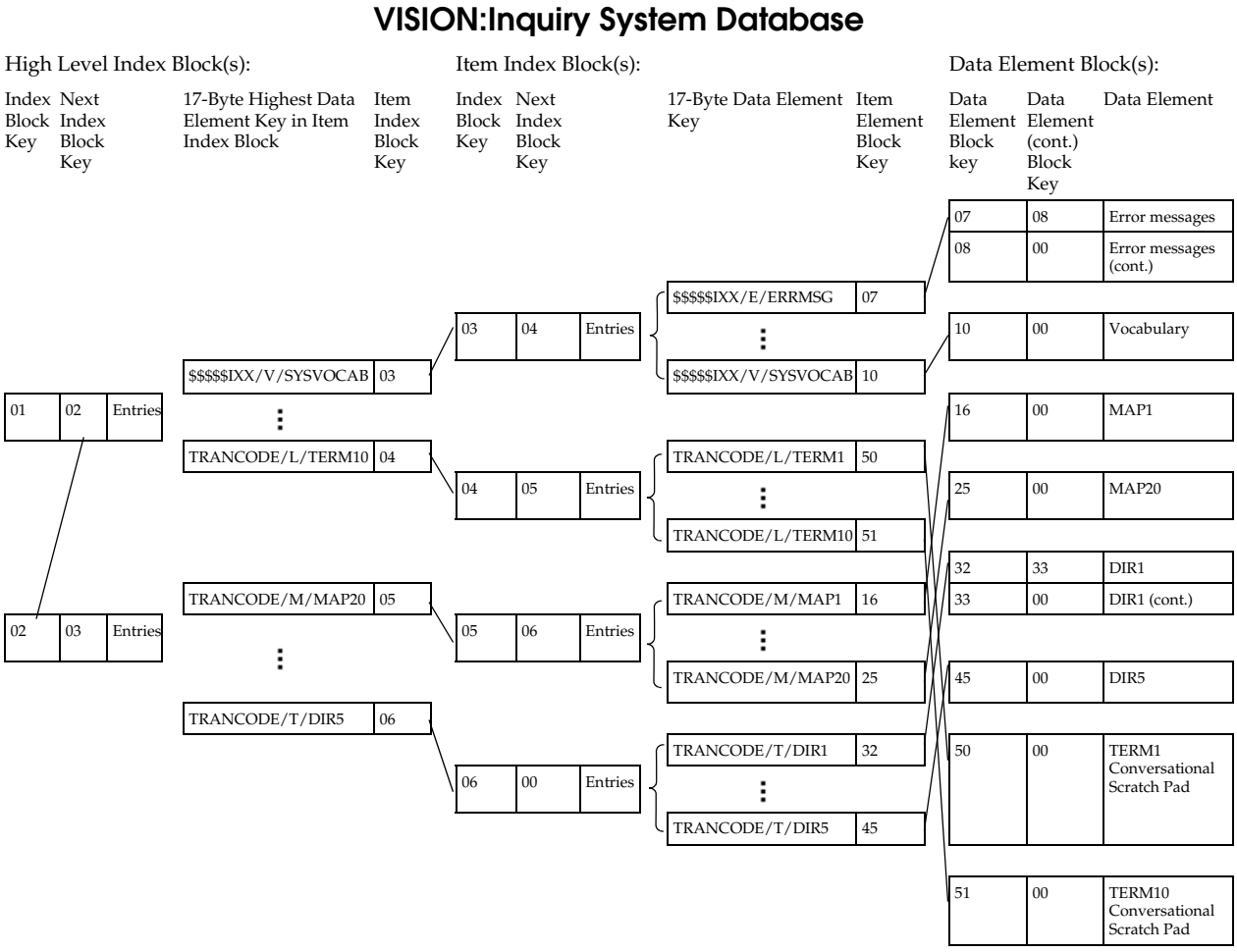


Figure 3-1 VISION:Inquiry System Database

Figure 3-1 shows:

- A system database containing two blocks of high level index with keys 1 and 2
- Item index blocks with keys 3 through 6
- Data element blocks with keys 7 through 51.

Trying to locate the data block for the map, MAP1 in the transaction code, TRANCODE, VISION:Inquiry searches the high level index block(s) sequentially to find the item index block key which contains the data element key, TRANCODE/M/MAP1 (05).

Note that the separator "/" is used for clarity and is not part of the key. Because the high level index contains the highest key of each item index block, the search stops when VISION:Inquiry finds a data element key equal to or greater than the requested data element key. It then searches the item index block with the key of 05 sequentially to find the data element block key for the element, MAP1 (16).

One item index entry is required for each of the following:

- The system error messages
- The standard system vocabulary
- Each database within an application
- Each directory within an application
- Each logical or generic terminal within an application

The item index entry also contains the block number within the directory block area for each database element.

Database Directory Block

The directory blocks area of the system database is also given a fixed size (number of blocks) when the initialization program IIINIT is executed. Each directory block can contain a maximum of 2,032 bytes of data.

The directory blocks are maintained in either a used or free state. The state of a directory block is determined by the status of a bit in a bit map.

As system database maintenance, inquiry definitions, function definitions, and deferred inquiries are performed, free space blocks are removed or added.

The System Vocabulary

The VISION:Inquiry vocabulary contains the words that are specified in inquiries as commands, quantifiers, arithmetic operators, logical operators, and noise words. Each vocabulary word equates to a stored code recognizable by the processor. During inquiry processing, VISION:Inquiry translates the inquiry words into sorted codes before they are executed.

The system must always contain at least one vocabulary. A standard vocabulary, SYSVOCAB, which is installed with the system, resides in application $$$$$IXX.

You can add synonyms to the standard system vocabulary to suit your specific installation requirements. You may find it necessary to create your own vocabularies, each with a unique identifier. These additional vocabularies can contain subsets of the system vocabulary with words and operators that only particular users, or groups of users, can access. The vocabulary can also be specified in languages other than English.

VISION:Inquiry Messages

Most messages issued by VISION:Inquiry reside in the system database. A standard set of these messages, used by the inquiry programs and the IIGEN utility, is distributed with the system to be stored in application $$$$$IXX.

By using the IIGEN utility, you can alter the content of these messages to meet your own installation requirements (such as a language other than English).

Only one set of messages may exist in a system database.

Some messages are hard-coded in the programs. The installation procedures in the *Advantage VISION:Inquiry for CICS Installation Guide* explain how these may be modified at installation time.

Most messages for the AQF facility of VISION:Inquiry reside in the BMS mapset IDBERRS. You can modify these messages, but they can contain only one line per message.

For additional information, see the *Advantage VISION:Inquiry Messages Guide*.

Database Maps

A user database is an IMS (DL/I) database, a DB2 table/view, or a VSAM data set from which VISION:Inquiry is to retrieve data in response to inquiries. A database map is the description of a user database used by VISION:Inquiry and stored in the system database. An appropriate database map must be generated before VISION:Inquiry can access a user database.

However, the generation of the database maps for DB2 tables accessed by native SQL syntax inquiries is not required in the system database.

Fields are the primary processing component of a user database. Each database map contains a description of all fields to be processed.

- For IMS (DL/I) databases, the segments, key fields, and hierarchy are also described in the map. These descriptions must match the description stored in the DBD. Non-key fields may be defined or redefined for any position and length within a segment and need not correspond to the DBD.
- For DB2 tables/views, each field represents a column.
- For VSAM data sets, the record length and key fields are described and must match the VSAM definition. Non-key fields may be defined or redefined for any position and length within the record.

Each database map uses one or more complete directory blocks. Database maps are maintained by the IIGEN utility.

User Directories

Database maps and user vocabularies are combined in user directories that are stored in the system database. Each user directory consists of one or more dedicated directory blocks within the database.

The directory is the profile of the VISION:Inquiry system. It provides a means for limiting access to database fields and vocabularies pointed to by specified terminals.

Users not requiring access to sensitive data can be denied access to that data. Restricted access is accomplished by excluding sensitive data fields from the database maps selected during directory generation. Similarly, selective vocabularies can be created with excluded functions and additional synonyms. The name of the selective vocabulary is specified during directory generation.

User directories are created by the IIGEN utility. Each directory may contain one or as many as 255 database maps and one vocabulary. In order to successfully create a directory, all of the database maps and the vocabulary comprising the directory must exist prior to its creation.

Once the directory is created, all of the vocabulary words and database field names are maintained in alphabetic sequence within the directory. When inquiries or functions are defined, they are converted to internal format and merged into the user directory in the correct alphabetic sequence.

The source of the defined inquiries is also written and kept in separate directory blocks which will be used by the Text Editor.

If definitions of inquiries and functions do not fit in the existing directory blocks, new user directory blocks will be obtained from the free space pool.

A directory must be created for the database map(s) needed by the user. However, individual directories can be created for combinations of database maps and unique vocabularies.

For example, if three database maps, A, B, and C, and vocabularies D and E are combined, at least two directories must be created. One directory contains database maps A, B, and C, and vocabulary D. The other contains database maps A, B, and C, and vocabulary E. In this type of configuration, if commands are excluded from the vocabulary, it is more efficient to exclude them from the vocabulary SYSVOCAB.

Connected Directories

The ability to connect directories is a feature of VISION:Inquiry that provides an additional level of flexibility in controlling access to data. Using this feature, you can connect directories in a list. When a database is specified in an inquiry, the directories are searched in the sequence they are connected.

Connected directories can contribute to a substantial saving of directory space by allowing you to define a map, a stored function, or a stored inquiry only once throughout the VISION:Inquiry system.

The following is a typical situation that illustrates how the implementation of connected directories is useful:

- Several databases are used for production
- Production users are not permitted to define stored inquiries or functions
- Most production inquiries are ad hoc rather than stored

- The systems group is the only one with the authority to define stored inquiries and functions
- Two programming teams are developing additional applications for the production environment
- The two programming teams can define stored inquiries and functions, but only the ones defined by the systems group are maintained on a permanent basis

You can only use connected directories with native VISION:Inquiry. Figure 3-2 on page 3-9 illustrates how directories can be generated for the environment described.

A directory is created for each of the two programming teams (IITST1) and (IITST2). These directories are created with a vocabulary and no maps. Both directories are connected to the system directory (IIASYS).

The IIBPRO directory is created for the production users. This directory contains a vocabulary without the DEFINE command. The system directory (IIASYS) contains the database maps. In this type of directory configuration, security is gained with a slight decrease in performance. This is because an additional database call is performed to obtain the required database map.

The IIASYS directory is created for the systems group. It contains a vocabulary and all maps. The directory is the Database Administrator (DBA's) directory.

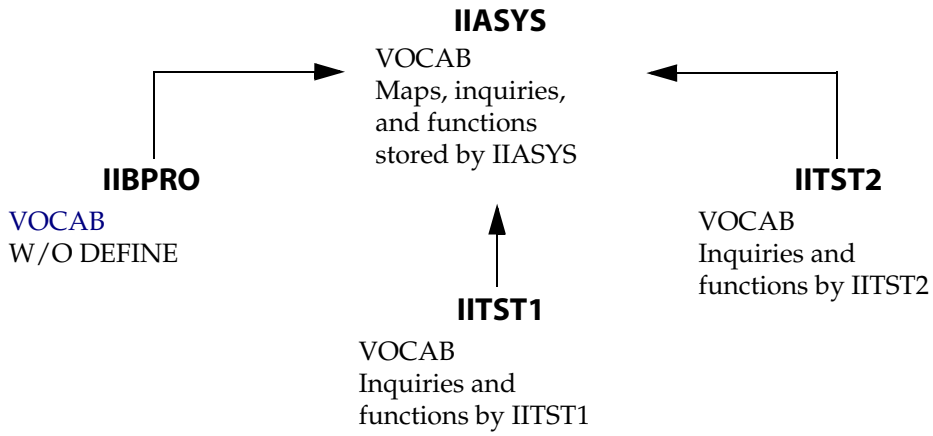


Figure 3-2 A Typical Connected Directory Configuration

Even though both IITST1 and IITST2 can define stored inquiries and functions, the system database load and unload utilities can be used to purge the database on a periodic basis. Through such purging, only the stored inquiries and functions that the systems group wants to keep remain on the database. These inquiries and functions are not available to the production directory.

Terminal Descriptions (TERMs)

TERMs and generic TERMs are the elements in the system database that describe the CICS terminals. The combination of the TERM or generic TERM and the CICS transaction code points to the directory; it is the directory that is used to locate the correct database map.

Directories, database maps, TERMs, and generic TERMs are linked to each other by applications. Applications are defined by the IIGEN utility. Each application can contain one terminal for a directory, or multiple terminals for each directory.

VISION:Inquiry terminal names must correspond to valid CICS operator IDs of users signing on at terminals. You can also change the system in such a way that VISION:Inquiry terminals correspond to valid CICS terminal IDs defined in the Terminal Control Table (TCT) or to valid CICS user IDs (refer to the *Advantage VISION:Inquiry for CICS Installation Guide* for more details). However, this is not true for generic terminal names (see Chapter 4, "The Definition Process"). The terminals that have access to the databases/VSAM files and vocabulary included in a directory are defined after generation of the directory.

Because terminals are defined within an application, each terminal is defined only once within that application. However, by using connected directories, each directory can be part of a different application and accessed from a single terminal.

TERMs and generic TERMs also provide VISION:Inquiry with information that can optimize its processing. For example, using such information, VISION:Inquiry can set limits for inquiry execution based upon the number of database calls issued and the number of logical pages displayed.

Conversational Scratch Pad Storage

This element of the database is used by VISION:Inquiry when it is necessary to checkpoint an inquiry due to certain types of interrupts. In these instances, checkpoint information is stored in the system database until processing can be resumed. A unique scratch pad area exists for each logical terminal defined within a transaction code.

Sort Work Data Set

The option of using a VSAM work data set for online sort processing is specified in the CICS parameter load module CVLCPARM. This VSAM KSDS data set is used as a work file to sort the output records.

The records, which are passed to the online sort routine, will be written to this data set and then sorted.

Automatic Query Facility (AQF) Components

AQF contains a set of menu screens (BMS maps) which provides you access data fields in the databases and generate queries, simply by selecting options from menus. These queries are executed by native VISION:Inquiry, and produce reports.

AQF also uses the CICS Temporary Storage Facility to keep track of the AQF screen options selected for a query. Using this information, AQF can rebuild the screens to allow changes to the previous query.

VISION:Journey VSAM Download Data Set

VISION:Journey uses a download data set during the transfer of data from the host to a PC file. This data set is a VSAM KSDS file with three different types of records, such as root, description, and data:

- The root record contains information for each extraction, such as subsequence number, the terminal that extracted the data, and the date and time the data was extracted.
- The description record contains information about each field of the extracted data; therefore the number of occurrences of description segments for each extraction is equal to the number of fields of the extracted data.
- The data record contains the actual data that should be downloaded to the PC file.

For a complete description of the fields of the download database segment, see Chapter 6, "Programming and Operation Considerations".

There are two more record types which are used internally by VISION:Journey. These two records, the Master Control Record and Master Header Record, are discussed in more detail in Chapter 6, "Programming and Operation Considerations".

VISION:Inquiry Test Data

This section describes VISION:Inquiry test data which are used to demonstrate VISION:Inquiry's capabilities and diagnosing problems. Two IMS (DL/I) databases (PLANT and SKILL), illustrated in *Figure 3-3 on page 3-12*, DB2 tables (Figure 3-4 on page 3-13), DB2 views (Figure 3-5 on page 3-14), and four VSAM data sets (two of which are pictured in Figure 3-6 on page 3-14), are part of the system database.

The test data serve two purposes. First, test data provide a tool with which you can learn the capabilities of VISION:Inquiry. Second, test data establish a common base between you and Computer Associates Technical Support, from which problems can be duplicated and identified and solutions developed.

IMS (DL/I) Test Databases

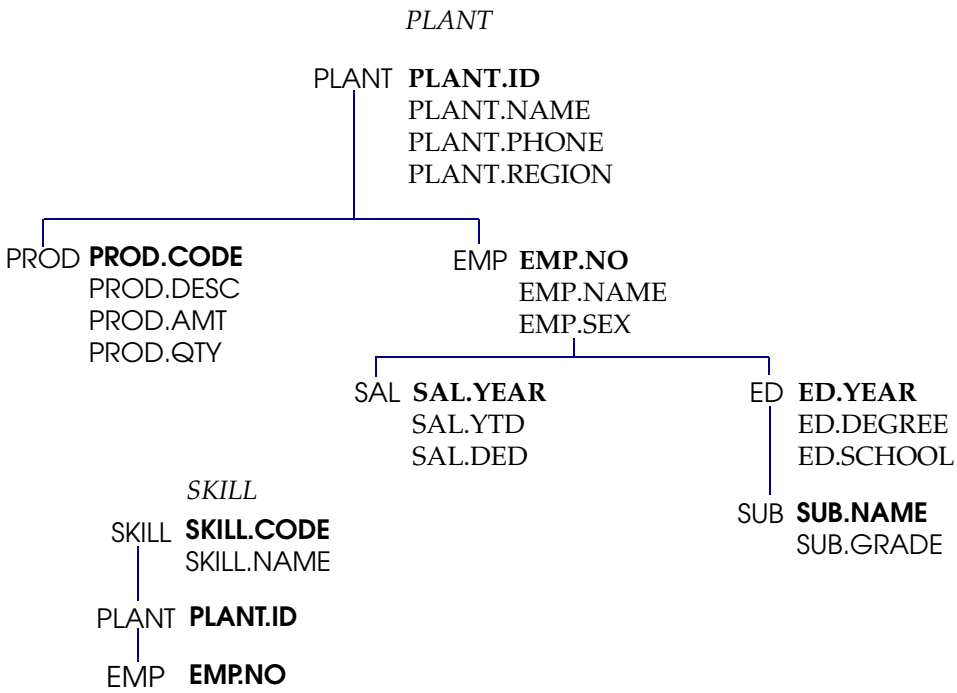


Figure 3-3 IMS (DL/I) Databases, PLANT and SKILL

Segment Key Fields for the PLANT database are PLANT.ID, PROD.CODE, EMP.NO, SAL.YEAR, ED.YEAR, and SUB.NAME. For the SKILL database, they are SKILL.CODE, PLANT.ID and EMP.NO.

DB2 Test Tables

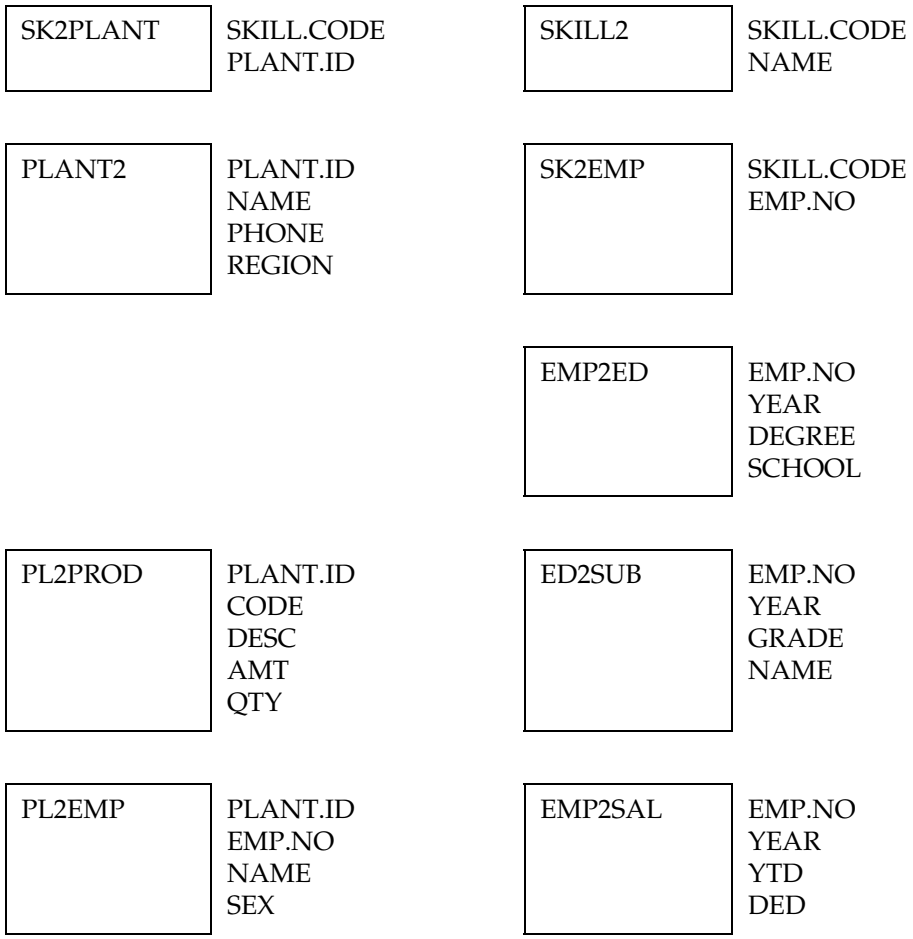


Figure 3-4 DB2 Test Tables

In Figure 3-4, the column names such as SKILL.CODE, PLANT.ID, or EMP.NO are those on which the tables are joined. These columns are used as matching fields when accessing multiple fields in VISION:Inquiry native mode and AQF (Automatic Query Facility).

DB2 Test Views

PRODUCTS	SALARIES	EDUCATION	SKILLS
PLANT_ID	PLANT_ID	PLANT_ID	SKILL_CODE
PLANT_NAME	PLANT_NAME	PLANT_NAME	SKILL_NAME
PLANT_PHONE	PLANT_PHONE	PLANT_PHONE	PLANT_ID
PLANT_REGION	PLANT_REGION	PLANT_REGION	EMP_NO
PROD_CODE	EMP_NO	EMP_NO	
PROD_DESC	EMP_NAME	EMP_NAME	
PROD_AMT	EMP_SEX	EMP_SEX	
PROD_QTY	SAL_YEAR	ED_YEAR	
	SAL_YTD	ED_DEGREE	
	SAL_DED	ED_SCHOOL	
		SUB_NAME	
		SUB_GRADE	

Figure 3-5 DB2 Test Views

VSAM Non-Hierarchical Test Data Sets

VSPLANT (KSDS)

VSPLANT.ID	VSEMP.NO	VSEMP.NAME	VSEMP.SEX	VSED.DEGREE	VSSAL.YTD	VSSAL.DED

Note: VSEMP.NO is a Key Field in the VSPLANT database, but not in the VSSKILL database.

VSSKILL (RRDS)

VSPLANT.ID	VSEMP.NO	VSSKILL.CODE	VSSKILL.NAME

Figure 3-6 VSAM Non-Hierarchical Data Sets, VSPLANT and VSSKILL

VSAM Hierarchical Test Data Sets

A VSAM hierarchical file structure is also supported in VISION:Inquiry. Using this structure means you can define your VSAM data sets with fixed or variable occurrences of data items to VISION:Inquiry. You can access this data online or batch through VISION:Inquiry, execute inquires, and create reports. Figure 3-7 is an example of a VSAM hierarchical record layout in COBOL. This record has both fixed and variable occurrences.

```
01   EMPLOYEE-RECORD.
       05 EMP-NO              PIC 9(5).
       05 EMP-NAME            PIC X(30).
       05 EMP-NUM-DEGREES     PIC 9.
          .
          .
          .
       05 EMP-DEGREE-DATA     OCCURS 1 TO 5 TIMES
          DEPENDING ON EMP-NUM-DEGREES.
          10 YEAR             PIC 99.
          10 SCHOOL           PIC X(20).
          10 DEGREE           PIC X(10).
       05 SAL-HISTORY         OCCURS 10 TIMES.
          10 SAL-YEAR         PIC 99.
          10 SAL-AMT          PIC S9(8)V99.
```

Figure 3-7 VSAM Hierarchical Record Layout (In COBOL)

VISION:Inquiry maps the VSAM hierarchical record layout in Figure 3-7 into the hierarchical structure in Figure 3-8. Fixed occurring segments always occur the same number of times for each record of the file. Variable occurring segments occur a different number of times based upon a field in the parent segment.

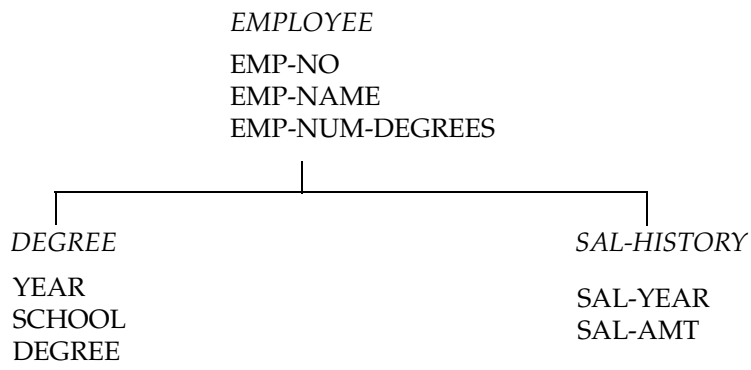


Figure 3-8 VSAM Hierarchical Record Layout (In VISION:Inquiry)

In Figure 3-8, the DEGREE segment is variable occurring based upon the field EMP-NUM-DEGREES. EMP-NUM-DEGREES must be in the DEGREE segment parent (EMPLOYEE segment).

Variable occurring segments are allowed for KSDS and ESDS files. Fixed occurring segments are allowed for KSDS, ESDS, and RRDS files.

4 The Definition Process

VISION:Inquiry requires that all IMS (DL/I) databases, DB2 tables/views, VSAM data sets, terminals, vocabularies, and directories be defined prior to entering any inquiries.

The only exception is the native SQL syntax facility for which you do not need to define DB2 tables/views to VISION:Inquiry.

The definition process is performed using the IIGEN utility which does the following:

- Builds database maps which describe, in detail, the format of IMS (DL/I) databases, DB2 tables/views, and VSAM data sets, and assigns names by which their elements are called in inquiries.
  This chapter contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing DB2 is specifically applicable to DB2 licensed sites.
- Builds user vocabularies which specify the commands and keywords used in inquiries
- Builds directories which collect the above-listed elements together
- Describes the terminals that use VISION:Inquiry and associates each with a directory
- Maintains the information listed above within applications, each of which represents the data accessible through one transaction code
- Stores text for most error messages issued by IIGEN and the inquiry program.

This chapter describes the IIGEN statements needed to define and maintain database maps, directories, terminals, and applications. The remaining functions of IIGEN (vocabulary and error message generation) and its JCL requirements are described in Chapter 5, "The Utilities".

Note: Throughout this guide, terms such as database, database map, and user database, should be understood to refer equally to IMS (DL/I) databases, DB2 tables/views, and VSAM data sets, unless explicitly qualified.

Defining the Directory

The various components required to access databases/VSAM files are combined by means of the directory definition. The directory acts as a table of contents for VISION:Inquiry. When VISION:Inquiry processes an application, it accesses a particular directory using a combination of the application name and the directory name.

Each directory can contain one or more database/VSAM file maps and one or more terminal descriptions. At the terminal level, the directory definition combines the map or portions of maps with the system or user vocabulary. (System and user vocabularies are discussed in Chapter 3, "System Components".)

The only exception is the native SQL syntax facility for which the DB2 tables and views descriptions are not required to be included in the directory and the information for the DB2 tables and views are extracted directly from the DB2 catalog.

The directory also provides a means of restricting access to specific data fields (not applicable to the native SQL syntax facility) and VISION:Inquiry features. Selected fields and vocabulary words can be excluded when the directory is defined, thus preventing a particular group of terminals from referencing the excluded field or vocabulary word. This is a primary security feature of the system.

Additional flexibility can be obtained through the use of the connected directory feature of VISION:Inquiry. This feature allows maps and stored functions or inquiries to be stored only once in the VISION:Inquiry system database.

DIRECTORY Control Statement Group

Directories must exist prior to generating TERMs. The IIGEN DIRECTORY control statement group defines directories. The following items are specified in the DIRECTORY statement:

- Directory name
- Database/VSAM file map name(s) (not applicable to the native SQL syntax facility)
- Vocabulary name
- Connected directory name
- Transaction code associated with the connected directory
- Maximum number of database calls for SORT.

Eliminating Fields and Vocabulary

The IIGEN EXCLUDE statement eliminates database/VSAM file fields (not applicable to the native SQL syntax facility) and vocabulary words from the directory. The following items are specified in the EXCLUDE statement:

- Database/VSAM file map in which to exclude a field
- Vocabulary in which to exclude a word
- Name of the field or word to exclude
- Name of the segment to exclude.

Defining VISION:Inquiry

IIGEN is the utility used to define the VISION:Inquiry interface with its system components.

Using the IIGEN Utility

Based upon the input commands and control statements specified, the IIGEN utility creates entries in the system database for maps, directories, and terminal descriptions (TERMs).

IIGEN Functions

The following are IIGEN statements and their functions:

MAPGEN Defines an IMS (DL/I) database, a DB2 table/view, or a VSAM data set to VISION:Inquiry and stores the information in the system database. This step is not required for DB2 tables and views accessed by the native SQL syntax facility.

DIRECTORY Defines user directories. This function combines a single map database, several maps, or even a portion of a map, with a system or a user vocabulary, subsequently creating a library entry accessible by VISION:Inquiry. (See Chapter 3, "System Components", for a description of maps.)

For each user, the created entries describe the functions that can be performed and the data upon which the user can operate.

The DIRECTORY function can also combine directories for efficiency and flexibility.

UPDATEDIR Updates user directories. This function updates the vocabulary and directory parameters of an existing directory. You can also add or update maps in an existing directory without recreating it using this function.

EXCLUDE Provides system security. By specifying (in an EXCLUDE statement) the names of database/VSAM file maps, system vocabularies, data fields, or vocabulary words, access to the specified information is restricted.

TERM Defines the terminals that may be accessed through each directory entry. This is another security capability built into the system. Through the specification or non-specification of terminals, access to certain terminals is authorized or denied. Because terminals point to directories, this is also a way to control the end user view. Using generic TERMs, a group of TERMs may be defined with one statement.

SYSTEM/ SYSLOAD Pertains to the selection and generation of the VISION:Inquiry vocabulary. Multiple system vocabularies can be specified with these functions.

DELETE Deletes elements such as maps, directories, TERMs, and vocabularies from the system databases.

ERRLOAD/ ERRMSG Allows error and diagnostic messages to be generated. Through these functions, error messages can be customized to a user's particular requirements.

MSGLIST Produces a formatted list of error and diagnostic messages.

IIGEN Control Statement Groups

IIGEN processes 80-byte input control statements that comprise the input data set. Each input data set contains a group of statements that must begin with an APPL statement and end with an END statement.

- The APPL statement must appear first in the input stream, except when an ERRLOAD statement is specified (see Chapter 5, "The Utilities", for details on ERRLOAD).
- For each execution of IIGEN, one or multiple APPL/END definitions, with their associated control statement groups, can be specified. The name parameter on the APPL statement is the IMS transaction code that is invoked in the inquiry.
- Except for APPL and END statements, which function together, all other control statements are specified with accompanying detail statements. These are referred to as control statement groups, and they occur between the APPL and END statements.
- The control statement that identifies the group must appear first; it is followed by all the detail statements for the group and is terminated by a FINISH statement.

The IIGEN control statement groups are:

- MAPGEN
- DIRECTORY
- UPDATEDIR
- TERM
- SYSTEM
- ERRLOAD
- DELETE

IIGEN Names

IIGEN refers to two types of names: those used by VISION:Inquiry, and those used by database managers and VSAM.

System Names

All system names used by IIGEN, VISION:Inquiry, database managers, and VSAM to access and describe data and structures, should be unique and must be eight characters or less in length. Although it is possible to have a database/VSAM file map named M1, a DBD/ddname named M1, and a system vocabulary also named M1, it is not desirable.

Field Names

All field names that you use in VISION:Inquiry must satisfy the VISION:Inquiry rules for uniqueness. In addition, they must not exceed the maximum length permitted for an application, as described with the NAMELENGTH parameter of the APPL statement. Since a default name length of 32 characters is applied, it is not necessary to specify the parameter explicitly unless you intend to expressly limit names to less than 32 characters.

IIGEN Coding Rules

When preparing command input statements for IIGEN, abide by the following coding conventions:

- Each statement must contain a command, with no embedded blanks, that identifies the statement type.
- Each statement may contain keyword parameters. These parameters may begin in any column to the right of the blank following the command (except on a continue statement). They may be in any order but must be separated by commas with no embedded blanks.
- Each parameter must be coded once, unless stated otherwise.
- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments may be placed after the last parameter with an intervening blank.
- Columns 1 through 71 of the input are scanned; columns 72 through 80 are ignored.
- An asterisk (*) in column 1 signifies that the entire statement is a comment. Comments may be entered at any point in the input.
- A number 1 in column 1 causes the page to eject when the output is listed.

Notation

The following notations are used in describing the various command statements:

- Brackets [ ] are used to indicate that the enclosed items are optional and may be omitted.
- Braces { } are used to group related items that are alternatives. One item must be chosen. The context indicates when choosing more than one item is appropriate.
- A default value within a group of items is underlined.
- The commands, keyword parameters, and default values are shown in upper-case letters.
- Generic descriptions of items you must provide are shown in lower-case letters.

Defining the Application

The IIGEN APPL statement defines the VISION:Inquiry application and its name to IIGEN. APPL must be the first statement in the IIGEN input stream except when the ERRLOAD statement is specified. (See Chapter 5, "The Utilities", for further information about ERRLOAD.)

APPL Statement

The APPL statement informs IIGEN that all the statements that follow it, up to the END statement, are related to the transaction code named in the statement. For online IMS access, the name specified in the APPL statement must be the same as the IMS transaction code for the MPP (Message Processing Program) and the BMP (Batch Message Processing) program. The APPL statement is specified in conjunction with the END statement.

The format of the APPL statement is:

APPL NAME=name(,NAMELENGTH=nn)

The APPL Statement parameters are

NAME=name Specifies the 1 to 4 character CICS transaction ID that identifies the logical application for which IIGEN is to create a VISION:Inquiry environment.

The APPL "name" for the vocabulary and the batch version can be up to eight characters.

NAMELENGTH =nn Specifies an optional decimal integer constant with the range of 1 to 32 that establishes the maximum permitted length of all field names in this application.

- If it is not present, a default value of 32 characters is established.
- If you omit this parameter, and the name you use is greater than 32 characters, truncation from the 33rd character occurs.

END Statement

The END statement must appear last in the input stream. It notifies IIGEN that there are no more control or detail statements in the input stream.

The format of the END statement is:

END

No parameters are required. Comments may appear in the parameter portion if desired. Every APPL statement must have an accompanying END statement.

FINISH Statement

In addition to the END statement, a similar statement known as FINISH must be specified at the end of each control statement group. The FINISH notifies IIGEN that all the detail statements for the particular control statement group are processed.

The format of the FINISH statement is:

FINISH

IIGEN input stream

A sample IIGEN input stream is illustrated in Figure 4-1.

See the control library members II.TCVLCNTL (IMSELEM), II.TCVLCNTL (VSAMELEM), and II.TCVLCNTL (DB2ELEM) for examples of the JCL to execute IIGEN.

```
APPL NAME=A

         MAPGEN MAP=MAP1,....
           SEGMENT SEGM=SEG1,....
             FIELD NAME=S1FLD1,....
             FIELD NAME=S1FLD2,....
           SEGMENT SEGM=SEG2,....
             FIELD NAME=S2FLD1,....
             FIELD NAME=S2FLD2,....
         FINISH

         DIRECTORY  NAME=DIR1,MAP=MAP1,....
         FINISH
         TERM  NAME=TRM1,DIRECTORY=DIR1,....
         FINISH
         TERM  NAME=TRM*,DIRECTORY=DIR1....
         FINISH

END
```

Figure 4-1 Sample IIGEN Input

Defining the Databases

This section and its related subsections are not applicable to the native SQL syntax facility and can be skipped for DB2 tables or views accessed by the facility.

MAPGEN Control Statement Group

The MAPGEN control statement group is used to describe the user databases to VISION:Inquiry. Each group (referred to as a "MAPGEN") creates one database map that defines one IMS (DL/I) database, DB2 table/view, or VSAM data set, to VISION:Inquiry. It does this by specifying the following:

- Database type (IMS, DB2, VSAMESDS, VSAMKSDS, VSAMRRDS, VSMHESDS, VSMHKSDS, or VSMHRRDS)
- Information needed to access the database, such as the DBD name for an IMS (DL/I) database, the authorization ID and table name for a DB2 table or view, or the ddname and password for a VSAM data set
- Field names
- Length of the field

- Field type
- Scaling information
- Field starting location
- Hierarchy
- Key field
- IMS secondary index fields
- VSAM alternate index fields.

Within each application, there may be many database maps; each one must be uniquely named within the application, and there must not be any fields in any database in the application by the same name. Each group begins with a MAPGEN statement that names the map to be stored in the system database and specifies information needed to access the user database. Each group is terminated by a FINISH statement.

The same user database may be redefined by as many different database maps as desired.

Specifying a MAPGEN Group for IMS (DL/I) Databases

Specifying a MAPGEN for an IMS (DL/I) database is similar to specifying a DBDGEN. Standard IMS structuring conventions must be followed when specifying the MAPGEN parameters. For instance, if a subordinate segment is defined, its parent must also be defined. The major difference between the DBDGEN and the MAPGEN is that the DBD need only qualify segments and key fields; the MAPGEN must describe in detail all fields to be accessed by VISION:Inquiry.

SEGMENT Statements

SEGMENT statements are coded to describe to VISION:Inquiry the segment hierarchy as defined for IMS.

- One SEGMENT statement must appear in the MAPGEN for each segment specified in the PSBGEN to which VISION:Inquiry is to be sensitive. That is, for every segment declared sensitive in the PSBGEN, a SEGMENT statement and, if applicable, an associated key field description must be specified.
- The SEGMENT statements that describe the segment type in a database must be entered in the same hierarchical sequence as they appear in the PSBGEN. VISION:Inquiry is confined to the limitations of IMS for defining hierarchical structures. No more than 15 levels of dependency can be defined, including the root segment, and no more than 255 segment types can be defined.

FIELD Statements

Each SEGMENT statement has FIELD statements associated with it. A FIELD statement describes one field within the segment to IIGEN.

- All FIELD statements specified between two SEGMENT statements, or between a SEGMENT and a FINISH statement, are associated with the previous SEGMENT
- When sensitivity is not required, it is valid to define segments without FIELD statements to VISION:Inquiry.

The following statements compose a MAPGEN for an IMS (DL/I) database.

```
MAPGEN statement
   SEGMENT statement
        FIELD statement
        FIELD statement
   SEGMENT statement
        FIELD statement
FINISH
```

Specifying a MAPGEN Group for DB2 Tables and Views

Note: This section does not apply to the native SQL syntax facility.

The MAPGEN for DB2 tables and views consists of only the MAPGEN/FINISH pair and FIELD statements:

- The MAPGEN statement specifies the information needed to access the entire table or view
- Each FIELD statement references a column.

The following illustrates a MAPGEN for a DB2 table or view.

```
MAPGEN statement
    FIELD statement
    FIELD statement
    FIELD statement
FINISH
```

Specifying a MAPGEN Group for VSAM Non-Hierarchical Data Sets

The MAPGEN for a VSAM non-hierarchical data set consists of the MAPGEN/FINISH pair, one RECORD statement, and FIELD statements:

- The MAPGEN statement specifies the information needed to access the data set.
- The RECORD statement states the type and length of records in the data set.
- The FIELD statements define the fields of the record that VISION:Inquiry can process.

The following illustrates a MAPGEN for a VSAM non-hierarchical data set.

```
MAPGEN statement
   RECORD statement
     FIELD statement
     FIELD statement
     FIELD statement
FINISH
```

Specifying a MAPGEN Group for VSAM Hierarchical Data Sets

The MAPGEN for a VSAM hierarchical data set consists of the MAPGEN/FINISH pair, one RECORD statement, SEGMENT statements, and FIELD statements:

- The MAPGEN statement contains the information needed to access the data set.
- The RECORD statement contains the type and length of records in the data set.
- The SEGMENT statements define the hierarchical structure of the data.
- The FIELD statements define the fields of the segments that VISION:Inquiry can process.

The following shows a MAPGEN group for a VSAM hierarchical data set.

```
MAPGEN statement
   RECORD statement
     SEGMENT statement
        FIELD statement
        FIELD statement
     SEGMENT statement
        FIELD statement
        FIELD statement
        FIELD statement
FINISH
```

The statements used in the MAPGEN group are described in the following pages.

MAPGEN Statement

The MAPGEN statement begins the specification of a database map and specifies information related to the entire IMS (DL/I) database, DB2 table/view, or VSAM data set.

The formats of the MAPGEN statement for the various database types are:

For IMS (DL/I) databases:

```
MAPGEN    (DBTYPE=IMS,)MAP=map-name,NAME=name,
          ({DESCRIPT|DESC}="text",)
          DBD=dbd-name(,OFFSET={1|pcb-number})
```

For DB2 tables and views:

```
MAPGEN    DBTYPE=DB2,MAP=map-name,NAME=name,
          ({DESCRIPT|DESC}="text",)
          AUTHID=auth-id,TABLENAME=table-name
```

For VSAM non-hierarchical data sets:

```
MAPGEN    DBTYPE={VSAMESDS|VSAMKSDS|VSAMRRDS},
          MAP=map-name,NAME=name,
          ({DESCRIPT|DESC}="text",)
          FILE=dd-name(,PASSWORD=read-pswd)
```

For VSAM hierarchical data sets:

```
MAPGEN    DBTYPE={VSMHESDS|VSMHKSDS|VSMHRRDS},
          MAP=map-name,NAME=name,({DESCRIPT|DESC}="text",)
          FILE=dd-name(,PASSWORD=read-pswd)
```

The MAPGEN statement parameters are:

DBTYPE=IMS	Default. Specifies that this map is for an IMS (DL/I) database.
DBTYPE=DB2	Specifies that this map is for a DB2 table or view.
DBTYPE=VSAMESDS	Specifies that this map is for a VSAM ESDS (Entry Sequenced Data Set).
DBTYPE=VSAMKSDS	Specifies that this map is for a VSAM KSDS (Key Sequenced Data Set).
DBTYPE=VSAMRRDS	Specifies that this map is for a VSAM RRDS (Relative Record Data Set).

DBTYPE=VSMHESDS Specifies that this map is for a VSAM hierarchical ESDS (Entry Sequenced Data Set).

DBTYPE=VSMHKSDS Specifies that this map is for a VSAM hierarchical KSDS (Key Sequenced Data Set).

DBTYPE=VSMHRRDS Specifies that this map is for a VSAM hierarchical RRDS (Relative Record Data Set).

MAP=map-name Specifies a 1 to 8 character name that is used to refer to this map in a DIRECTORY statement. This name must be unique within the containing APPL group.

NAME=name Specifies the 1 to 8 character name by which this database is called in an inquiry.

DESCRIPT="text" or DESC="text" Provides up to 60 characters of description for the database. If the description contains blanks, commas, or parentheses, it must be enclosed in double quotation marks, for a maximum of 62 characters including the double quotation marks. The description cannot span more than one line.

- The description can be displayed by the native VISION:Inquiry PDDDS and PDD WHOLE commands or by the AQF (Automatic Query Facility) V (for view description) command.
- The description is also displayed when using the Intraccess option.

DBD=dbd-name Specifies the DBD name by which the database is known to IMS.

OFFSET= {1|pcb-number} Designates which PCB is to be used to access the database when the inquiry PSB contains more than one PCB for this DBD name. The first PCB with the specified DBD name is designated as OFFSET=1, the second OFFSET=2, and so on. The default for this parameter is OFFSET=1.

AUTHID=auth-id Specifies the DB2 authorization ID of the table or view.

TABLENAME= table-name Specifies the name of the table or view.

FILE=dd-name Specifies the VSAM data set ddname to which this data set is assigned by the batch region JCL or the CICS file entry.

PASSWORD= read-pswd	Specifies the read password, if required, for the VSAM data set.

SEGMENT Statement for IMS (DL/I) Databases

The SEGMENT statement describes the name, type, length, key, and hierarchical position of a segment of an IMS (DL/I) database.

The format of the SEGMENT statements is:

```
SEGMENT   SEGM=segment-name,PARENT={0|parent-segment},
          (KEY=key-field-name,(KEYLEN=key-field-length,))
          (TYPE=V,)(TWIN=NO,)BYTES=segment-length
```

The SEGMENT statement parameters are:

SEGM= segment-name	Specifies the IMS name of the segment as described in the DBDGEN.
PARENT=0	Default. Indicates the root segment. This must be specified or defaulted for the first, and only the first, SEGMENT statement within a MAPGEN for an IMS (DL/I) database.
PARENT= parent-segment	Specifies the segment name of this segment's logical parent, defined by a previous SEGMENT statement in this MAPGEN group.
KEY=key-field-name	Specifies the IMS name of this segment's key field. This is the name specified within the DBD on the FIELD statement for the KEY FIELD. Code this parameter only if the segment is keyed.
KEYLEN= key-field-length	Specifies the length in bytes, of the segment's key field. This parameter need not be coded unless the key field will not be defined to VISION:Inquiry by one or more FIELD statements (for example, for security reasons). Otherwise, specify the KEY or KEY and KEYPOS parameters on the FIELD statement(s) for the key field.
TYPE=V	Indicates that this is a variable length segment.
TWIN=NO	Indicates that this segment occurs at most once under a given parent segment; therefore, VISION:Inquiry need not issue a DL/I call to retrieve a second occurrence.

BYTES= segment-length	Specifies the length (or, for variable length segments, the maximum length), in bytes, of the segment.

RECORD Statement for VSAM Data Sets

The RECORD statement specifies the type and length of records in a VSAM data set.

The format of the RECORD statement is:

```
RECORD BYTES=record-length(,TYPE=V)
```

The RECORD statement parameters are

BYTES=record-length	Specifies the length (or, for variable length records, the maximum length), in bytes, of a record within the VSAM data set.
TYPE=V	Indicates that the data set contains variable length records.

SEGMENT Statement for VSAM Hierarchical Data Sets

The SEGMENT statement describes the name, length, and hierarchical position of a segment of a VSAM hierarchical data set.

The format of the SEGMENT statement is:

```
SEGMENT    SEGM=segment-name,PARENT={0|parent-segment},
               BYTES=segment-length
               (,OCRFLD=occurs-depending-on-field-name|
               ,OCRNUM=number-occurrences)
```

The SEGMENT statement parameters are:

SEGM=segment-name	Specifies a 1 to 8 character name.
PARENT=0	Default. Indicates the root segment. This must be specified or defaulted for the first, and only the first, SEGMENT statement within a MAPGEN.
PARENT= parent-segment	Specifies the segment name of this segment's logical parent, defined by a previous SEGMENT statement in this MAPGEN group.

BYTES=segment-length	Specifies the length of the segment in bytes.
OCRFLD=occurs-depending-on-field-name	For a variable occurring segment the field which has the number of occurrences for this particular segment. ■ This name cannot be more than eight characters and must be the first name specified on the FIELD statement when it was defined. ■ This field must have been previously defined in this segment's parent segment.
OCRNUM=number-occurrences	Defines the number of times this segment occurs. This defines a fixed occurring segment.

FIELD Statement

The FIELD statement describes a single data item. Each FIELD statement relates to one field of a segment in an IMS (DL/I) database, one column of a DB2 table or view, or one field of a record in a VSAM data set.

On the following three pages are examples of FIELD statement syntax for DL/I databases, DB2 tables and views, and VSAM hierarchical and non-hierarchical data sets.

In each FIELD statement example, large brackets have been added to illustrate the optional portions of the statements. These brackets are NOT part of the actual statement syntax.

The format of the FIELD statement is:

For DL/I databases:

```
FIELD LENGTH=field-length,
      START=location,

TYPE=C,
   or
TYPE={B | X},
     (SCALE={0 | decimal-places},)
     (OUTEDIT={FULL | ZERO | NONE},)
   or
TYPE={N | P},(INEDIT=NOSIGN,)
     (SCALE={0 | decimal-places},)
     (OUTEDIT={FULL | ZERO | NONE},)
   or
TYPE=Y,SUBSTR=(start-digit,number-of-digits),
     (SCALE={0 | decimal-places},)
     (OUTEDIT={FULL | ZERO},)
   or
TYPE={U | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9},
     (OUTLTH=output-length,)

KEY={SEQ-U | SEQ-M | EQUAL},
     (KEYPOS=position-in-key,)
   or
KEY=SRCH,
     (KEYPOS=position-in-key,)
     (DBNAME=IMS-field-name,)
   or
KEY=INDX,
     (KEYPOS=position-in-key,)
     (DBNAME=IMS-field-name,)
TARGET=target-segment,
OFFSET=pcb-number,

({DESCRIPT | DESC}="text",)
```

For DB2 tables and views:

```
FIELD LENGTH=field-length,

[
TYPE=C,
   or
TYPE={B|X},
(SCALE={0|decimal-places},)
(OUTEDIT={FULL|ZERO|NONE},)
   or
TYPE={N|P},(INEDIT=NOSIGN,)
(SCALE={0|decimal-places},)
(OUTEDIT={FULL|ZERO|NONE},)
   or
TYPE=Y,SUBSTR=(start-digit,number-of-digits),
(SCALE={0|decimal-places},)
(OUTEDIT={FULL|ZERO},)
   or
TYPE={U|0|1|2|3|4|5|6|7|8|9},
(OUTLTH=output-leNgth,)
(DBTYPE=SQL-data-type,)
(DBNAME=DB2-column-name,)
]

({DESCRIPT|DESC}="text",)
```

For VSAM hierarchical and non-hierarchical data sets:

```
FIELD LENGTH=field-length,
      START=location,

[
TYPE=C,
   or
TYPE={B|X},
    (SCALE={0|decimal-places},)
    (OUTEDIT={FULL|ZERO|NONE},)
   or
TYPE={N|P},(INEDIT=NOSIGN,)
    (SCALE={0|decimal-places},)
    (OUTEDIT={FULL|ZERO|NONE},)
   or
TYPE=Y,SUBSTR=(start-digit,number-of-digits),
    (SCALE={0|decimal-places},)
    (OUTEDIT={FULL|ZERO},)
   or
TYPE={U|0|1|2|3|4|5|6|7|8|9},
[    OUTLTH=output-length,)

KEY=SEQ-U,
    (KEYPOS=position-in-key,)
   or
KEY=INDX,
    (KEYPOS=position-in-key,)
    (PATHNAME=dd-name,)
     (PASSWORD=read-password,)

(OCRCNT=YES,)
```

The FIELD statement parameters are:

LENGTH= field-length	Specifies the length of the field in bytes. The LENGTH may be from 1 to 255 bytes.
START=location	Specifies the starting location of the field within the segment or record (for example, the first byte of the segment or record is indicated by START=1). When defining a field of an IMS (DL/I) database for secondary indexing that is made up of non-contiguous fields within the segment, specify START=1. The field is taken from the key feedback area in the PCB, and the START parameter is ignored.
TYPE=C	Default. Indicates a character field.
TYPE={B|X}	Indicates a signed binary integer field. The LENGTH may be from 1 to 4 bytes.
TYPE=N	Indicates a numeric character (zoned decimal) field. The LENGTH may be from 1 to 15 bytes (digits).
TYPE=P	Indicates a packed decimal field. The LENGTH may be from 1 to 8 bytes (1 to 15 digits).
TYPE=Y	Indicates a subfield of a packed decimal field. The LENGTH may be from 1 to 8 bytes (1 to 15 digits). A subfield may not be a key field (that is, the KEY parameter must not be specified).
TYPE= {U|0|1|2|3|4|5|6|7|8|9}	Indicates a field to be handled by a user conversion exit. The LENGTH may be from 1 to 255 bytes.
INEDIT=NOSIGN	Specifies for a numeric character (zoned decimal) or packed decimal field, that value is stored on the database without a sign (that is, the “sign bits” are “1111” rather than ”1100”/“1101” for plus/minus signed data).
SUBSTR= (start-digit, number-of-digits)	Specifies, in packed decimal digits, the starting location and length of a subfield within a packed decimal field.
SCALE= decimal-places	Specifies an optional scaling factor that describes the position of an assumed decimal point in the decimal representation of the numeric field relative to the rightmost position. A scale factor of zero is assumed if the parameter is not coded. As an example, a “dollar and cents” field has a scale of +2 (decimal point two places from the rightmost position).

OUTEDIT=FULL
Default. Specifies full numeric field editing on output: decimal point placement, comma insertion, and zero suppression.

OUTEDIT=ZERO
Specifies limited numeric field editing on output: decimal point placement and zero suppression.

OUTEDIT=NONE
Specifies no numeric field editing on output. OUTEDIT=NONE cannot be specified for substring fields (TYPE=Y).

OUTLTH= output-length
Specifies the number of bytes required for output of the field, as returned from a user conversion exit. For example, a field processed by a user conversion exit returning a calendar date of the form "MM-DD-YY" would require OUTLTH=8. If the OUTLTH parameter is not specified, the output length is assumed to be equal to the field length specified by the LENGTH parameter.

KEY=SEQ-U
Indicates that this field is the key field of the segment or record. For an IMS (DL/I) database and VSAM KSDS data set, the field is a unique key. This field must always be in the root segment for a VSAM hierarchical data set.

KEY=SEQ-M
Indicates that this field is the key field of the containing segment. The field is a non-unique key. This field must always be in the root segment for a VSAM hierarchical data set.

KEY=EQUAL
Indicates that this field is the key field of the root segment of an HDAM database.

KEY=SRCH
Indicates that this field is an IMS search field.

KEY=INDX
Indicates that this field is the key field of an IMS secondary index or a VSAM alternate index.

- This field must always be in the root segment for a VSAM hierarchical data set.
- The KEY parameter is not valid for ESDS and RRDS VSAM data sets.

KEYPOS= position-in-key
Indicates that this field comprises only part of the key of the segment or record and specifies the position within the key (first byte = 1) where this field begins.

DBNAME= IMS-field-name

Specifies, for IMS (DL/I) database fields with KEY=SRCH or KEY=INDX, the name by which the field is known to IMS (the 1 to 8 character field name used in the DBD). If this parameter is not coded for a KEY=SRCH or KEY=INDX field, the first 8 characters of the first NAME parameter are used.

DBNAME= DB2-column-name

Specifies, for fields within a MAPGEN for a DB2 table or view, the 1 to 18 character name of the column this field represents. If this parameter is not coded for a field, the first 18 characters of the first NAME parameter are used.

TARGET= target-segment

Specifies, for a field in an IMS (DL/I) database map with KEY=INDX, the target segment name for the secondary index field; that is, the name specified in the LCHILD parameter of the secondary index DBD.

OFFSET= pcb-number

Indicates, for a field in an IMS (DL/I) database where KEY=INDX is specified, which PCB is to be used to access the secondary index. The first PCB with the DBD name specified in the DBD parameter of the MAPGEN statement is designated as OFFSET=1, the second OFFSET=2, and so forth.

DBTYPE= SQL-data-type

Specifies the internal format of a field in a MAPGEN for a DB2 table or view to be processed by a user conversion exit.

SQL-data-type must be coded as an unsigned decimal integer; VISION:Inquiry will place this value in the SQLTYPE field of the SQLDA as the data type of the host variable when retrieving data for the associated column from DB2. (For example, to process a floating point column of a DB2 table using a conversion exit, specify TYPE=U or 0-9 and DBTYPE=480 on the FIELD statement describing the column.)

PATHNAME= dd-name

Specifies, for a field designated as KEY=INDX within a MAPGEN for a VSAM data set, the ddname of the path which relates the alternate index to the base cluster. If this parameter is omitted for a KEY=INDX field in a MAPGEN for a VSAM data set, the first 8 characters of the first NAME parameter are used.

PASSWORD= read-password

Specifies, for a field designated as KEY=INDX within a MAPGEN for a VSAM data set, the read password, if required, for a path that relates the alternate index to the base cluster for batch processing.

DESCRIPT="text" or DESC="text"

Provides up to 60 characters of description for the field. If the description contains blanks, commas, or parentheses, it must be enclosed in double quotation marks, for a maximum of 62 characters including the double quotation marks.

- The description cannot span more than one line.
- The description can be displayed by the native VISION:Inquiry PDDDS and PDD WHOLE commands or by the AQF (Automatic Query Facility) V (for view description) command.

NAME=name

Specifies a name by which this field is called in an inquiry.

- This parameter may be specified from 1 to 8 times on one FIELD statement.
- The maximum length of name is determined by the NAMELENGTH parameter of the APPL statement for the application in which the containing database map appears.

OCRCNT=YES

Indicates that this field is a count field for a dependent variable occurs segment. This field contains the actual number of occurrences for the dependent segment and is used for VSAM hierarchical data sets only.

MAPGEN Group Notes

- The name specified by the NAME parameter of a MAPGEN statement must not be the same as any name specified in the NAME parameter on any FIELD statement, or any other MAPGEN statement, of any map to be included in the same directory.
- The names specified by the NAME parameter of FIELD statements must be unique within a map, but may be duplicated within different maps.
- Neither field nor database names may be the same as any vocabulary word to be included in the same directory.
- For the most part, IIGEN cannot verify the correctness or consistency of your field definitions. You must be sure they are correct.
- Considerable flexibility is provided for defining overlapping fields, concatenated fields, and so on. For instance, there is no restriction on the number of overlapping field definitions or the manner in which they overlap. If you are redefining complete or partial fields that differ in name and characteristics, you may define them on multiple FIELD statements within the segment, record, or map in which they occur.
- If synonyms for a key field are required, they may be defined without the KEY parameter.
- Specifying too many synonyms (multiple FIELD statements defining the same field or multiple NAME parameters on a single FIELD statement) can have an adverse effect on performance, since the size of the application within the system database, and the number of database calls necessary to retrieve it are increased. Faster system database access is attained if frequently used root key fields are given names near the beginning of the EBCDIC collating sequence.
- KEYPOS is used when a key is made up of more than one field, which you want to define individually. For example, if a key is made up of two fields called “department” and “cost,” and you want to define them separately, code:

```
FIELD START=1,LENGTH=2,TYPE=C,NAME=DEPT,KEY=SEQ-U,KEYPOS=1
FIELD START=3,LENGTH=4,TYPE=P,NAME=COST,KEY=SEQ-U,KEYPOS=3
```

- When TYPE=Y and SUBSTR are coded for the field, the field is defined as a subfield. A subfield is a string of packed decimal digits within a larger packed field. It differs from an ordinary packed field in two ways:
  - It can begin and end on any half-byte boundaries, not only on byte boundaries.
  - The last half-byte represents the low-order packed decimal digit; there is no sign.

For example, suppose a 4-byte packed field DATE contains a calendar date in the form YYMMDD (for example, May 30, 1995 is stored as 950530), and you want separate access to the month, day, and year. The record looks like this:

1	2	3	4	5	6	7	8	(digit positions)
0	9	5	0	5	3	0	C	(field contents)

and the required MAPGEN statements are as follows:

```
FIELD LENGTH=4,START=1,TYPE=P,NAME=DATE
FIELD LENGTH=4,START=1,TYPE=Y,SUBSTR=(2,2),NAME=YEAR
FIELD LENGTH=4,START=1,TYPE=Y,SUBSTR=(4,2),NAME=MONTH
FIELD LENGTH=4,START=1,TYPE=Y,SUBSTR=(6,2),NAME=DAY
```

Using the COBOL Converter

The COBOL converter helps in creating field definitions for your VSAM data set or IMS (DL/I) database. The COBOL converter accepts special SEGMENT statements for IMS (DL/I) or RECORD statements for VSAM data sets; it then generates IIGEN acceptable field definition statements from COBOL copy books in MVS Partitioned Data Set (PDS), CA-Librarian, and CA-Panvalet libraries.

- For IMS (DL/I) databases, key field and search field information is derived from the corresponding DBD load module and added to the appropriate field definition statements. The secondary index field information for an IMS (DL/I) database is also derived from the corresponding DBD and PSB load modules and added to the appropriate field definition statements.
- For VSAM KSDS data sets, the key field information is derived from the VSAM catalog and added to the appropriate field definition statement if the cluster parameter is coded in the record statement (discussed later in this chapter).

The COBOL converter reads the MAPGEN and SEGMENT/RECORD statements and generates IIGEN field definition statements to the file with the ddname SYS004. The following JCL is necessary to execute the COBOL converter. After executing the step, you can make any necessary changes to the SYS004 output file and then use it as input to IIGEN.

Note: The COBOL copy book referenced in the samples do not really exist, but can be created if needed.

II.TCVLCNTL (IMSCOBL)

The control library member II.TCVLCNTL (IMSCOBL) contains sample JCL for execution of the COBOL converter program for use with IMS (DL/I) databases.

```
//* JCL TO EXECUTE COBOL CONVERTER FOR DL/I DATABASES
//*   - THIS JOB IS RUN TO CREATE THE IIGEN FIELD DEFINITION STATEMENTS
//*     FROM COBOL COPYBOOKS.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES TO MATCH THOSE YOU WILL USE.
//*   - VERIFY, ADD, OR CHANGE ANY IIGEN INPUT DATA OR COMMENTS AS SYSIN
//*     DATA WHICH YOU WANT TO BE PASSED TO THE OUTPUT DATA SET(SYS004)
//*   - THE MAPGEN AND SEGMENT STATEMENTS SHOWN IN THIS SAMPLE AS SYSIN
//*     DATA ARE ONLY EXAMPLES AND THE COPY MEMBERS SPECIFIED IN THE
//*     COPYCOBOL PARAMETERS DON'T EXIST. CHANGE THE MAPGEN AND SEGMENT
//*     STATEMENTS TO MATCH THE DATABASE AND COPY MEMBERS YOU WANT TO
//*     USE.
//*
//STEP01   EXEC PGM=INQCOBCV,REGION=400K
//STEPLIB  DD DSN=YOUR.VISION:Inquiry.LOAD.LIBRARY,DISP=SHR
//         DD DSN=IMSDBD.LIBRARY,DISP=SHR
//         DD DSN=IMSPSB.LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSCOPY  DD DSN=YOUR.COBOL.COPYBOOK.LIBRARY,DISP=SHR
//SYS004   DD DSN=MAPS,DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,2)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD *
*    sample input data
APPL NAME=IQIO
*
*    COBOL DEFINITION EXAMPLE FOR PLANT
*
MAPGEN MAP=IIDMMAP,DBD=IIDBDDM,NAME=PLANT
 SEGMENT SEGM=PLANT,COPYCOBOL=COBPLANT,PSB=INQIO
 SEGMENT SEGM=PROD,COPYCOBOL=COBPROD
 SEGMENT SEGM=EMP,COPYCOBOL=COBEMP
 SEGMENT SEGM=SAL,COPYCOBOL=COBSAL
 SEGMENT SEGM=ED,COPYCOBOL=COBED
 SEGMENT SEGM=SUB,COPYCOBOL=COBSUB
FINISH
*
*    COBOL DEFINITION EXAMPLE FOR SKILL
*
MAPGEN MAP=IIDSMAP,DBD=IIDBDDS,NAME=SKILL
 SEGMENT SEGM=SKILL,COPYCOBOL=COBSKILL,PSB=INQIO
 SEGMENT SEGM=PLANT,COPYCOBOL=COBSKPLT
 SEGMENT SEGM=EMP,COPYCOBOL=COBSKEMP
FINISH
END
```

Figure 4-2 JCL to Execute the COBOL Converter for IMS (DL/I) Databases

Notes:

- The output data set (SYS004) can either be a sequential data set or a PDS member.
- The IMS DBD library must contain a DBD load module for each database to be defined in the MAPGEN statements.
- The PSB parameter in the SEGMENT statement is optional and only used for secondary index fields. For secondary index fields, the IMS PSB library must contain the PSB load module with the secondary index PCB(s). Note that the PSB parameter is needed in the first SEGMENT only.

- If you do not have any secondary indices in your database, the PSB parameter and the DD statement for the PSB library are not needed and can be deleted.
- If the COBOL copy members are in a CA-Panvalet library, include:

```
//PANDD1  DD DSN=YOUR.CA-PANVALET.LOAD.LIBRARY,DISP=SHR
```

- If the COBOL copy members are in a CA-Librarian library, include:

```
//MASTER  DD DSN=YOUR.LIBRARIAN.LOAD.LIBRARY,DISP=SHR
```

- The COBOL copy book referenced in the sample does not really exist, but can be created if needed.

II.TCVLCNTL (VSAMCOBL)

The control library member II.TCVLCNTL (VSAMCOBL) contains sample JCL for execution of the COBOL converter program for use with VSAM data sets.

```
//* JCL TO EXECUTE COBOL CONVERTER FOR VSAM DATASETS
//*   - THIS JOB IS RUN TO CREATE THE IIGEN FIELD DEFINITION STATEMENTS
//*     FROM COBOL COPYBOOKS.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES TO MATCH THOSE YOU WILL USE.
//*   - VERIFY, ADD, OR CHANGE ANY IIGEN INPUT DATA OR COMMENTS AS SYSIN
//*     DATA WHICH YOU WANT TO BE PASSED TO THE OUTPUT DATA SET(SYS004)
//*   - THE MAPGEN AND RECORD STATEMENTS SHOWN IN THIS SAMPLE AS SYSIN
//*     DATA ARE ONLY EXAMPLES AND THE COPY MEMBER SPECIFIED IN THE
//*     COPYCOBOL PARAMETER DOES NOT EXIST. CHANGE THE MAPGEN AND RECORD
//*     STATEMENT TO MATCH THE VSAM DATASET AND COPY MEMBERS YOU WANT
//*     TO USE.
//*
//STEP01   EXEC PGM=INQCOBCV,REGION=400K
//STEPLIB  DD DSN=YOUR.VISION:Inquiry LOAD.LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSCOPY  DD DSN=YOUR.COBOL.COPYBOOK.LIBRARY,DISP=SHR
//SYS004   DD DSN=MAPS,DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(TRK,(5,2)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN    DD *
*    sample input data
APPL NAME=IQIO
*
*    COBOL DEFINITION EXAMPLE FOR VSPLANT
*
MAPGEN DBTYPE=VSAMKSDS,MAP=VSPLMAP,NAME=VSPLANT,FILE=VSPLDS
 RECORD BYTES=80,COPYCOBOL=COBPLANT
FINISH
END
```

Figure 4-3 JCL to Execute the COBOL Converter for VSAM Non-Hierarchical Files

Notes:

- The output data set (SYS004) can either be a sequential data set or a PDS member.
- If the COBOL copy members are in a CA-Panvalet library, include:

  ```
  //PANDD1  DD DSN=YOUR.CA-PANVALET.LOAD.LIBRARY,DISP=SHR
  ```

- If the COBOL copy members are in a CA-Librarian library, include:

  ```
  //MASTER  DD DSN=YOUR.LIBRARIAN.LOAD.LIBRARY,DISP=SHR
  ```

COBOL Converter Input Parameters for DL/I

Comments can be included at any point in the input and are identified by an asterisk (*) in column 1. All comments are passed to the SYS004 output data set and accepted as comments in the VISION:Inquiry IIGEN utility.

Any statement acceptable to IIGEN, whether or not used by the COBOL converter, can be included in the job stream. For example, an APPL statement can be included before the first MAPGEN statement.

A FINISH statement should be inserted after the last SEGMENT statement in the MAPGEN.

After the maps are defined, you may include DIRECTORY, TERM definitions, and an END statement; all of these statements and the FINISH statement are passed, as is, to the SYS004 data set.

To create a definition for a database, the MAPGEN statement must be followed by all SEGMENT statements for the database definition.

The format of the MAPGEN statement is:

```
MAPGEN MAP=IIDMMAP,DBD=IIDBDDM,NAME=PLANT
```

The COBOL interface retrieves the specified DBD load modules, for use with the SEGMENT statements, to build portions of the SEGMENT and FIELD statements.

The format of the SEGMENT statement is:

```
SEGMENT SEGM=PLANT
```

No other information should be included on the SEGMENT statement except the COPY COBOL information and the optional PSB parameter if you need secondary index parameters to be generated.

The converter retrieves the BYTES, TYPE, and PARENT information from the corresponding SEGM statement in the DBD. If the DBD name from the MAPGEN is not found, a message is issued, and the SEGMENT is left as specified.

The key information on the SEGMENT statement is built if a FIELD parameter in the DBD is identified as a KEY for the segment.

The FIELD statements for the definitions are built from the COBOL field definitions. COBOL definitions are associated with a SEGMENT statement in one of the following three ways:

- Specify a COBOL copy book from a PDS as follows:

```
SEGMENT SEGM=PLANT,COPYCOBOL=COBPLANT,PSB=INQIO
```

The PDS must contain a member named COBPLANT. All the FIELD statements for VISION:Inquiry definitions are built from the various 01, 05, and so on, levels of COBOL statements. The field type, length, location, and name will be built, and all dashes (-) become periods (.). Key fields are indicated from information in the DBD. A listing of the copied members is produced; if you wish to suppress this listing, specify:

```
COPYCOBOL=(COBPLANT,NOPRINT)
```

Note: The PSB parameter should be coded for the first SEGMENT statement only.

The PSB parameter in the SEGMENT statement is optional. If your database contains secondary indices, the necessary FIELD statement parameters (that is, KEY=INDX, TARGET, DBNAME, and OFFSET) will be built from information in the DBD and PSB. The secondary index parameters will not be built if the PSB parameter is not coded, and a warning message will be issued.

- Include COBOL definitions instream as follows:

```
SEGMENT SEGM=PLANT,PSB=INQIO
$COBOL
   COBOL statements
$ECOBOL
```

The same information is created as specified in the first example above.

- Include a COBOL definition from CA-Panvalet or CA-Librarian as follows:
For CA-Panvalet:

```
SEGMENT SEGM=PLANT,COPYPCOBOL=COBPLANT,PSB=INQIO
```

For CA-Librarian:

```
SEGMENT SEGM=PLANT,COPYLCOBOL=COBPLANT,PSB=INQIO
```

The same information is created as specified in 1 above. If you want the listing of copied fields suppressed, specify:

```
COPYLCOBOL=(COBPLANT,NOPRINT),PSB=INQIO
```

or

```
COPYPCOBOL=(COBPLANT,NOPRINT),PSB=INQIO
```

COBOL Converter Input Parameters for VSAM

Comments can be included at any point in the input and are identified by an asterisk (*) in column 1. All comments are passed to the SYS004 output data set and accepted as comments in the VISION:Inquiry IIGEN utility.

Note: This feature is only for VSAM non-hierarchical data sets.

Any statement acceptable to IIGEN, whether or not used by the COBOL converter, can be included in the job stream. For example, an APPL statement can be included before the first MAPGEN statement.

A FINISH statement should be inserted after the last RECORD statement in the MAPGEN.

After the maps are defined, you may include DIRECTORY, TERM definitions, and an END statement; all of these statements and the FINISH statement are passed, as is, to the SYS004 data set.

To create a definition for a non-hierarchical VSAM data set, the MAPGEN statement must be followed by a RECORD statement for the file definition.

The format of the MAPGEN statement is:

```
MAPGEN      DBTYPE={VSAMKSDS | VSAMESDS | VSAMRRDS},
            MAP=VSPLMAP,FILE=VSPLDS,NAME=VSPLANT
```

The format of the RECORD statement is:

```
RECORD      BYTES=80(,TYPE=V)(,CLUSTER=dataset-name)
            (,PATH=(dd-name(,password)))
```

The RECORD statement parameters are:

CLUSTER= dataset-name	Used for VSAM KSDS data sets only. Generates the KEY=SEQ-U parameter for the appropriate FIELD statement.

PATH= (dd-name,password)

Generate alternate index parameters for the appropriate FIELD statement. You can have more than one PATH parameter if multiple alternate indices are defined for your VSAM data set.

The ddname specifies the PATH ddname for the alternate index, and the password is required only if your PATH is password protected.

- If the password is not used, the enclosing parentheses on ddname may be deleted.
- If the PATH parameter is used, the DD statement for the PATH should be added in the COBOL converter JCL.
- If you do not code the PATH parameter, the alternate index parameters will not be generated.
- The PATH parameter is not needed if you do not have the alternate index for your VSAM KSDS/ESDS data set.

Notes:

- If you do not code the cluster parameter for KSDS data sets, the COBOL converter issues a warning message and continues processing. In this case, it is the user's responsibility to add the KEY=SEQ-U parameter to the appropriate field statement.
- The parameters for the RECORD statement can be continued on the next line. A comma following the last parameter on the RECORD statement causes a continuation to the next line.
- The FIELD statements for the definitions are built from the COBOL field definitions. COBOL definitions are associated with a RECORD statement in one of the following three ways.
- Specify a COBOL copy book from a PDS as follows:

```
RECORD BYTES=80,CLUSTER=VS.VSPLDS,COPYCOBOL=COBPLANT
```

The PDS must contain a member named COBPLANT. All the FIELD statements for VISION:Inquiry definitions are built from the various levels of COBOL statements (01, 05, and so on). The field type, length, location, and name will be built, and all dashes (-) become periods (.). A listing of the copied members is produced; if you wish to suppress this listing, specify:

```
COPYCOBOL=(COBPLANT,NOPRINT)
```

- Include COBOL definitions instream as follows:

```
RECORD BYTES=80
$COBOL
   COBOL statements
$ECOBOL
```

  The same information is created as specified in the first example above.

- Include a COBOL definition from CA-Panvalet or CA-Librarian as follows:

  For CA-Panvalet:

```
RECORD BYTES=80,CLUSTER=VS.VSPLDS,COPYPCOBOL=COBPLANT
```

  For CA-Librarian:

```
RECORD BYTES=80,CLUSTER=VS.VSPLDS,COPYLCOBOL=COBPLANT
```

  The same information is created as specified in the first example above. If you want the listing of copied fields suppressed, specify:

```
COPYLCOBOL=(COBPLANT,NOPRINT)
```

  or

```
COPYPCOBOL=(COBPLANT,NOPRINT)
```

COBOL Features Not Supported

The following items in COBOL statements are not correctly converted or flagged as errors by the converter:

- 66 levels
- 88 levels
- OCCURS clauses
- Numeric fields with more than 9 digits to the right of the decimal point
- Binary fields larger than S9(9)
- The P edit parameter on the PICTURE clause always generates a zero SCALE value
- Non-unique COBOL field names are flagged as errors

VISION:Inquiry does not perform any validation of COBOL statements. The COBOL statements should have been processed by the COBOL compiler prior to their use in creating VISION:Inquiry definitions.

IMS Considerations

The current version of the COBOL interface supports all primary database definitions, but it does not support logical DBDs.

TWIN=NO is not generated for SEGMENT statements, when only one occurrence of the segment exists.

If the database type is HDAM, KEY=EQUAL is specified for the root key field.

KEY=INDX, TARGET, DBNAME, and OFFSET parameters on the FIELD statement are generated for secondary indices if you specify a PSB parameter in SEGMENT statement. However, the COBOL converter feature does not support the secondary index, which is formed of more than one field (that is, SRCH parameter of DBD).

OFFSET is not generated on MAPGEN statements.

Logical DBDs are not supported by the COBOL converter.

KEYPOS information for generic keys is not specified by the COBOL converter.

TYPE U, 0-9, X, or Y fields need to be coded separately, because the COBOL converter cannot create them.

OUTLTH, OUTEDIT, and SUBSTR information is not coded on the FIELD statement.

KEY=SRCH and IMSNAME information is coded for all non-key fields defined in the DBD.

KEYLEN is not generated on the SEGMENT statement.

VSAM Considerations

For VSAM KSDS data sets, KEY=SEQ-U is generated for the key FIELD statement if the CLUSTER parameter is coded in the RECORD statement.

Note: The current version does not support VSAM hierarchical data set structures.

KEY=INDX, PATHNAME, and PASSWORD parameters on the FIELD statement are generated if the PATH parameter is coded in the RECORD statement.

KEYPOS information for generic keys is not specified by the COBOL converter.

TYPE U, 0-9, X, or Y fields need to be coded separately because the COBOL converter cannot create them.

OUTLTH, OUTEDIT, and SUBSTR information is not coded on the FIELD statement.

User Responsibility

After the COBOL converter generates IIGEN field definitions, it is the responsibility of the user to peruse the SYS004 data set. Changes should be made to this data set to add and refine the information, especially the information the COBOL converter was unable to deduce from the data available to it. The items mentioned in COBOL Features Not Supported on page 4-35, IMS Considerations on page 4-36, and VSAM Considerations on page 4-37, must be considered as manual changes to the SYS004 data set when appropriate.

Using the DB2 Catalog Program

This section and its related subsections are not applicable to the native SQL syntax facility and can be skipped for DB2 tables or views accessed by the facility.

The DB2 catalog program generates MAPGEN, FIELD, FINISH, and END statements for your DB2 tables to a file with the ddname of DB2OUT.

- This file can be either a sequential data set or a partitioned data set member.
- This program accepts DB2MAP statements and generates IIGEN MAPGEN and FIELD statements for the DB2 definition of your table.

The following JCL is necessary to execute the program. After executing it, you can make any necessary changes to the DB2OUT output file and add the required APPL statement for input to IIGEN.

II.TCVLCNTL (DB2CATL)

The control library member II.TCVLCNTL (DB2CATL) contains the sample JCL to execute the DB2 catalog program.

```
//* JCL TO EXECUTE THE DB2 CATALOG PROGRAM FOR DB2 TABLES
//*   - THIS JOB IS RUN TO CREATE THE IIGEN FIELD DEFINITION STATEMENTS
//*     FOR DB2 TABLES.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES TO MATCH THOSE YOU WILL USE.
//*   - VERIFY OR CHANGE THE DB2 CATALOG LOAD MODULE NAME, DB2CATA
//*     TO MATCH THE MODULE NAME YOU HAVE GENERATED.
//*   - VERIFY AND CHANGE THE SYSTEM, PLAN, AND LIB PARAMETERS TO MATCH
//*     THOSE YOU WILL USE.
//*   - ADD ANY COMMENTS AS SYSIN DATA WHICH YOU WANT TO BE PASSED TO
//*     TO THE OUTPUT DATA SET(DB2OUT).
//*
//STEP1    EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=400K
//STEPLIB  DD DSN=DSN510.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330,LRECL=133)
//DB2OUT   DD DSN=DB2.MAPS,DISP=(,CATLG),UNIT=SYSDA,
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=FB,BLKSIZE=23440,LRECL=80)
//SYSTSIN  DD *
DSN SYSTEM(YOUR DB2 SUBSYSTEM ID)
RUN  PROGRAM(DB2CATA) PLAN(YOUR PLAN NAME) -
     LIB('YOUR LOAD LIBRARY')
END
//SYSIN    DD *
*  SAMPLE INPUT DATA
*
DB2MAP CATLID=SYSIBM,
       TABLE=IIPLANT,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IIPLANT SAMPLE TABLE",
       MAP=MAPPL,
       AUTHID=DYLINQ
DB2MAP CATLID=SYSIBM,
       TABLE=IIPLANT_PROD,
       IPTYPE=C,
```

Figure 4-4 JCL to Execute the DB2 Catalog Program (Page 1 of 2)

```
       DESCRIPT="DESCRIPTION FOR IIPLANT_PROD SAMPLE TABLE",
       MAP=MAPPLPR,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IIPLANT_EMP,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IIPLANT_EMP SAMPLE TABLE",
       MAP=MAPPLEM,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IIEMP_SAL,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IIEMP_SAL SAMPLE TABLE",
       MAP=MAPEMSA,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IIEMP_ED,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IIEMP_ED SAMPLE TABLE",
       MAP=MAPEMED,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IIEMP_ED_SUB,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IIEMP_ED_SUB SAMPLE TABLE",
       MAP=MAPEMEDS,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IISKILL,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IISKILL SAMPLE TABLE",
       MAP=MAPSK,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IISKILL_PLANT,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IISKILL_PLANT SAMPLE TABLE",
       MAP=MAPSKPL,
       AUTHID=DYLINQ
*
DB2MAP CATLID=SYSIBM,
       TABLE=IISKILL_EMP,
       IPTYPE=C,
       DESCRIPT="DESCRIPTION FOR IISKILL_EMP SAMPLE TABLE",
       MAP=MAPSKEM,
       AUTHID=DYLINQ
/*
```

Figure 4-4 JCL to Execute the DB2 Catalog Program (Page 2 of 2)

DB2MAP Statement - Input Parameters for the DB2 Catalog Program

The DB2MAP statement identifies the DB2 table for which MAPGEN and FIELD statements are to be generated. It also provides information about your DB2 subsystem.

The format of the DB2MAP statement is:

```
DB2MAP   AUTHID=authid,TABLE=tablename(,IPTYPE=iptype)
         (,CATLID=catlid)(,MAP=mapname)(,NAME=name)
         (,DESCRIPT=text)
```

The DB2MAP statement parameters are:

AUTHID=authid — The authorization ID (high-level qualifier) of the table or view.

TABLE=tablename — The name (low-level qualifier) of the table or view.

IPTYPE=iptype — Indicates whether the MAPGEN should be generated using the DB2 system catalog or the user table itself. Allowable values are "C" for the system catalog and "D" for the user table. The default value is "D".

CATLID=catlid — The authorization ID (high-level qualifier) of the DB2 system catalog when using "IPTYPE=C". The default for this field is SYSIBM.

MAP=mapname — A 1 to 8 character internal name assigned to the table map. The default is the first 8 characters of the DB2 table name.

NAME=name — A 1 to 8 character name used to reference this table in an inquiry. If this field is not input, the DB2 LABEL field for the table is used. If it is not present, the first 8 characters of the DB2 table name are used.

DESCRIPT=text — The description of the table.

- If it contains blanks, commas, or parentheses, it must be enclosed with double quotation marks for a maximum length of 62 characters including the double quotation marks. The description cannot span more than 1 line.
- If this field is not input, the DB2 REMARKS field for the table is used. If it is not present, no description is generated.

The description can be displayed by the PDDDS and PDD WHOLE commands or by the AQF (Automatic Query Facility) V (for View description) command.

Comments can be entered at any point in the input. They are identified by an asterisk (*) in column 1 or can appear on any other input record if preceded by at least 1 blank.

Output from the DB2 Catalog Program

The output is a report highlighting all errors and any default values used, as well as the MAPGEN and FIELD statements. The following is a list of error messages:

MSG #	SEV LVL	Message (Page 1 of 2)
DB20001	12	DB2MAP KEYWORD MISSING.
DB20002	12	END OF KEYWORD (=) NOT FOUND.
DB20003	08	INVALID KEYWORD.
DB20004	08	COMMA OR BLANK NOT FOUND AT END THE RECORD.
DB20005	08	AUTHID IS A REQUIRED FIELD.
DB20006	08	TABLE IS A REQUIRED FIELD.
DB20007	08	NO " AT END OF FIELD.
DB20008	04	IPTYPE INVALID OR MISSING, DEFAULTS TO D.
DB20009	04	EXPECTED CONTINUATION NOT RECEIVED.
DB20010	04	CATLID INVALID OR MISSING, DEFAULTS TO SYSIBM.
DB20011	16	INTERNAL ERROR. CONTACT COMPUTER ASSOCIATES.
DB20012	04	MAP NAME NOT ENTERED. DEFAULTS TO 1ST 8 CHARS OF TABLE NAME.
DB20013	04	NAME NOT ENTERED. DEFAULTS TO 1ST 8 CHARS OF TABLE NAME.
DB20020	08	SQL ERROR HAS OCCURRED. MAPGEN NOT GENERATED.
DB20021	04	SQL ERROR HAS OCCURRED. WILL RETRY WITH IPTYPE=D.
DB20030	08	TABLE DOES NOT EXIST IN SYSIBM.SYSTABLES.
DB20031	08	TABLE DOES NOT EXIST IN SYSIBM.SYSCOLUMNS.
DB20099	xx	CONTROL CARD ERROR, MAPGEN NOT GENERATED.

MSG #	SEV LVL	Message (Page 2 of 2)
DB20198	00	MAPGEN AND FIELD STATEMENTS SUCCESSFULLY GENERATED.
DB20199	xx	CONTROL CARD VALID.
DB20299	00	HIGHEST SEVERITY LEVEL ENCOUNTERED = xxxx.
DB29999	16	INTERNAL ERROR. CONTACT COMPUTER ASSOCIATES.

Samples of Database Definitions

The following samples illustrate how to code MAPGENs for databases with various features such as secondary indexing, variable length segments, a non-hierarchical VSAM structure, an alternate key, a variable record, and a hierarchical VSAM structure.

Sample MAPGEN for a Database

Figure 4-5 illustrates a sample MAPGEN for a database.

```
*
*        DEPARTMENT MAP
*
 MAPGEN  MAP=DEPTMAP,DBD=DEPTDBD,NAME=DEPT
*
 SEGMENT SEGM=DEPARTMT,BYTES=99,PARENT=0,KEY=DEPTID
 FIELD NAME=DEPTID,LENGTH=3,START=1,KEY=EQUAL
 FIELD NAME=DEPTMGR,LENGTH=25,START=4
 FIELD NAME=DEPTNAME,LENGTH=25,START=29
 FIELD NAME=DEPTBDGT,LENGTH=15,START=54,TYPE=N
 FIELD NAME=DEPTDIV,LENGTH=3,START=69
 FIELD NAME=DEPTEXT,LENGTH=3,START=72,TYPE=P,KEY=SRCH,OUTEDIT=ZERO
 FIELD NAME=DEPTMGRN,LENGTH=5,START=75,TYPE=N,OUTEDIT=ZERO
*
 SEGMENT SEGM=PRODUCTS,BYTES=80,PARENT=DEPARTMT,KEY=PRODCODE
 FIELD NAME=PRODCODE,LENGTH=5,START=1,KEY=SEQ-U
 FIELD NAME=PRODNAME,LENGTH=15,START=6
 FIELD NAME=PRODDESC,LENGTH=25,START=21
 FIELD NAME=PRODCOST,LENGTH=4,START=46,TYPE=X,SCALE=+2
 FIELD NAME=PRODQNTY,LENGTH=3,START=50,TYPE=P
 FIELD NAME=PRODBDGT,LENGTH=8,START=53,TYPE=P
*
 SEGMENT SEGM=SECTIONS,BYTES=85,PARENT=DEPARTMT,KEY=SECID
 FIELD NAME=SECID,LENGTH=4,START=1,KEY=SEQ-U
 FIELD NAME=SECNAME,LENGTH=25,START=5
 FIELD NAME=SECMGR,LENGTH=25,START=30
 FIELD NAME=SECMGRN,LENGTH=3,START=55,TYPE=P,OUTEDIT=ZERO
 FIELD NAME=SECBDGT,LENGTH=8,START=58,TYPE=P
*
SEGMENT SEGM=PROJECTS,BYTES=85,PARENT=SECTIONS,KEY=PROJCODE
```

Figure 4-5 A Sample MAPGEN for a Database (Page 1 of 2)

```
 FIELD NAME=PROJCODE,LENGTH=5,START=1,KEY=SEQ-U
 FIELD NAME=PROJNAME,LENGTH=15,START=6
 FIELD NAME=PROJDESC,LENGTH=25,START=21
 FIELD NAME=PROJBDGT,LENGTH=15,START=46,TYPE=N
 FIELD NAME=PROJPROD,LENGTH=5,START=61
*
SEGMENT SEGM=STAFF,BYTES=48,PARENT=SECTIONS,KEY=STAFMBRN
 FIELD NAME=STAFMBRN,LENGTH=3,START=26,TYPE=X,OUTEDIT=ZERO,KEY=SEQ-U
 FIELD NAME=STAFMBR,LENGTH=25,START=1
*
 FINISH
*
*
```

Figure 4-5 A Sample MAPGEN for a Database (Page 2 of 2)

Sample MAPGEN with a Secondary Index

The following sample illustrates how to code the MAPGEN for a database with a secondary index.

```
*
 SEGMENT SEGM=SECTIONS,BYTES=85,PARENT=DEPARTMT,KEY=SECID
 FIELD NAME=SECID,LENGTH=4,START=1,KEY=SEQ-U
 FIELD NAME=SECNAME,LENGTH=25,START=5
 FIELD NAME=SECMGR,LENGTH=25,START=30
 FIELD NAME=SECMGRN,LENGTH=3,START=55,TYPE=P,OUTEDIT=ZERO
 FIELD NAME=SECBDGT,LENGTH=8,START=58,TYPE=P
 FIELD NAME=SECMGRX,LENGTH=25,START=30,OFFSET=2,KEY=INDX,
                    TARGET=DEPARTMT
```

Figure 4-6 A Sample of a Portion of a MAPGEN with a Secondary Index

Sample DBDGEN with the Same Secondary Index

The following example illustrates how to code the MAPGEN for a database with a variable length segment..

```
*
  SEGM  NAME=DEPARTMT,BYTES=99,PARENT=0
  FIELD NAME=(DEPTID,SEQ,U),BYTES=3,START=1
  FIELD NAME=DEPTMGR,BYTES=25,START=4
  FIELD NAME=DEPTEXT,BYTES=3,START=72,TYPE=P
  LCHILD NAME=(SECMGRX,DEPSMDBD),PTR=INDX
  XDFLD NAME=SECMGRX,SRCH=SECMGR,SEGMENT=SECTIONS,SUBSEQ=/SXSECT
```

Figure 4-7 A Sample of a Portion of a DBDGEN with the Same Secondary Index

Sample MAPGEN with a Variable Length Segment

```
*
 SEGMENT SEGM=SCHOOLS,BYTES=106,PARENT=PERSONAL,TYPE=V,KEY=SCHOOL
 FIELD NAME=SCHOOL,LENGTH=20,START=5,KEY=SEQ-U
 FIELD NAME=SCHADDR1,LENGTH=20,START=27
 FIELD NAME=SCHADDR2,LENGTH=20,START=47
 FIELD NAME=SCHADDR3,LENGTH=20,START=67
 FIELD NAME=SCHADDR4,LENGTH=20,START=87
 FIELD NAME=LINE.CNT,LENGTH=2,START=25,TYPE=P
*
```

Figure 4-8 A Sample of a Portion of a MAPGEN with a Variable Length Segment

Sample DBDGEN with a Variable Length Segment

```
*
 SEGM  NAME=SCHOOLS,BYTES=(106,27),PARENT=PERSONAL
 FIELD NAME=(SCHOOL,SEQ,U),BYTES=20,START=5
*
```

Figure 4-9 A Sample of a Portion of a DBDGEN with a Variable Length Segment

Sample VSAM Non-Hierarchical Definition for VSPLANT

```
*   VSPLANT MAP
*
 MAPGEN MAP=VSPLMAP,FILE=VSPLDS,NAME=VSPLANT,DBTYPE=VSAMKSDS
*
 RECORD BYTES=80
 FIELD START=1,LENGTH=5,TYPE=C,NAME=VSPLANT.ID
 FIELD START=6,LENGTH=5,TYPE=N,NAME=VSEMP.NO,KEY=SEQ=U
 FIELD START=11,LENGTH=25,TYPE=C,NAME=VSEMP.NAME
 FIELD START=36,LENGTH=1,TYPE=C,NAME=VSEMP.SEX
 FIELD START=37,LENGTH=2,TYPE=C,NAME=VSED.DEGREE
 FIELD START=39,LENGTH=7,TYPE=N,NAME=VSSAL.YTD,SCALE=+2
 FIELD START=46,LENGTH=7,TYPE=N,NAME=VSSAL.DED,SCALE=+2
*
 FINISH
*
```

Figure 4-10 Sample of VSAM Non-Hierarchical Definition for VSPLANT

Sample of Defining VSPLANT.ID as an Alternate Key

```
*
 FIELD START=1,LENGTH=5,TYPE=C,NAME=VSPLANT.ID,KEY=INDX,
 PATHNAME=ALTPATH
*
```

Figure 4-11 Sample of Defining VSPLANT.ID as an Alternate Key

Sample of Defining Variable Record in a MAPGEN

```
*
 RECORD BYTES=95,TYPE=V
 FIELD START=1,LENGTH=5,TYPE=C,NAME=VSPLANT.ID
 FIELD START=6,LENGTH=5,TYPE=N,NAME=VSEMP.NO,KEY=SEQ-U
 FIELD START=11,LENGTH=25,TYPE=C,NAME=VSEMP.NAME
 FIELD START=36,LENGTH=1,TYPE=C,NAME=VSEMP.SEX
```

Figure 4-12 Sample of Defining Variable Record in a MAPGEN

Sample of VSAM Hierarchical Definition for VSHPLANT

```
*        MAPGEN FOR PLANT SAMPLE VSAM HIERARCHICAL FILE
*
 MAPGEN  MAP=VSHPLMAP,FILE=VSHPLDS,NAME=VSHPLANT,DBTYPE=VSMHKSDS,
         DESC="VSMH TEST FILE - PRODUCTS AND EMPLOYEES"
*
 RECORD  BYTES=747,TYPE=V
*
 SEGMENT SEGM=VSHPLANT,PARENT=0,BYTES=41
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=VSHPLANT.ID,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=6,LENGTH=2,TYPE=C,NAME=VSHPLANT.REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   START=8,LENGTH=7,TYPE=C,NAME=VSHPLANT.PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   START=15,LENGTH=25,TYPE=C,NAME=VSHPLANT.NAME,
         DESC="NAME OF PLANT"
 FIELD   START=40,LENGTH=2,TYPE=B,NAME=PLEMPOCR,NAME=VSHPLANT.EMPOCR,
         OCRCNT=YES,DESC="NO. OF EMPLOYEE SEGMENTS PER PLANT"
```

Figure 4-13 Sample of VSAM Hierarchical Definition for VSHPLANT

Defining Directories

The DIRECTORY control statement is used to combine the various elements of the application and store the information in the system database.

In order to successfully create a directory, all the maps necessary for the creation of the directory must be previously defined by the MAPGEN statement (not applicable to the native SQL syntax facility). A directory cannot be created for maps that do not exist. You can define as many directories as necessary within an application. You may also specify connected directories for additional flexibility in controlling access to data by users.

DIRECTORY Control Statement Group

A DIRECTORY control statement group consists of the following statements:

- DIRECTORY statement
- EXCLUDE statement (optional)
- FINISH statement

Figure 4-14 illustrates a typical application directory configuration.

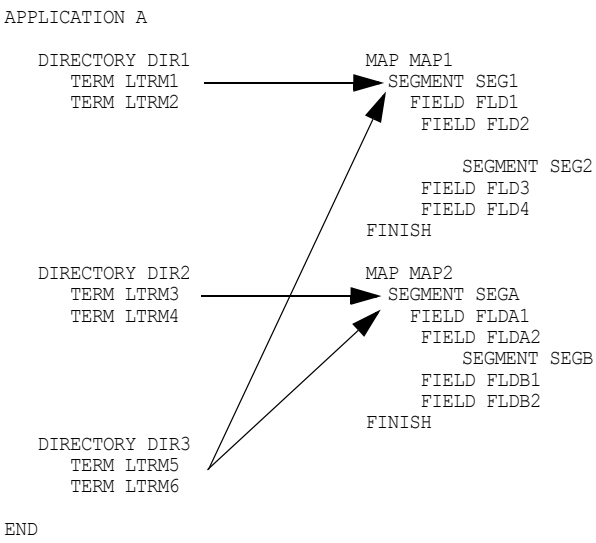


Figure 4-14 A Typical Application's Directory Configuration

DIRECTORY Statement

The format of the DIRECTORY statement is:

DIRECTORY NAME=name(,MAP=mapname)(,VOCAB=vocabname)
(,SLIMIT=num)
(,CTRAN=dirtran,CDIR=dirname)

The DIRECTORY statement parameters are:

NAME=name Defines legal name of up to eight characters to be appended to the directory and identifies it for later references by TERM statements.

MAP=mapname Defines name of a map that is used in the creation of this directory. This parameter may be used up to 255 times. The MAP parameter is not required for the DB2 tables and views accessed by the native SQL syntax facility.

VOCAB= vocabname Defines the name of a user vocabulary that was previously defined using the SYSTEM and NAME control statements, which are explained in Chapter 5, "The Utilities".

If the parameter is not present, the system vocabulary is used.
This parameter is optional and is only specified if the named vocabulary exists.

SLIMIT=num Specifies the maximum number of database calls or VSAM record reads allowed when SORT is specified by an online inquiry.

When sorting is requested in an inquiry, this limit replaces the TERM time limit (see TERM control statement, TIME parameter). Default is 5000 (maximum value is 32756). No limit if 0 is specified.

If the number of calls/reads exceeds this value, processing will terminate and an error message is issued. This parameter does not apply to VISION:Journey inquiries and should not be coded. This parameter is also applied to the "ORDER BY" command of the native SQL syntax facility.

CTRAN=dirtran Specifies the name of the transaction code to which this directory is connected. If this parameter is specified, the CDIR parameter must also be specified.

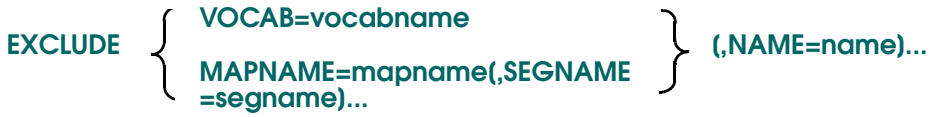
CDIR=dirname Specifies the directory name to which this directory is connected. If this parameter is specified, the CTRAN parameter must also be specified.

EXCLUDE Statement

The EXCLUDE statement is available for removing items from a directory when it is being generated. The EXCLUDE statement provides you with a means for securing and protecting sensitive data elements.

- Through specification of the EXCLUDE statement, you can omit sensitive elements from the directory at the time it is generated. EXCLUDE allows you to remove field or segment names from databases and from the MAPGEN, and statements from the system or user vocabulary.
- Specifying elements on an EXCLUDE statement eliminates them from the directory you are generating; it does not delete them from the map or vocabulary where they are stored. They still remain accessible to the users who have access to the other directories.
- The EXCLUDE statement is placed between a DIRECTORY and a FINISH statement. The DIRECTORY statement preceding the EXCLUDE must not be terminated by a comma; a comma results in the EXCLUDE being ignored.
- Only one EXCLUDE statement may be specified for each directory to remove statements from a vocabulary, or fields and segments from a map.

The format of the EXCLUDE statement is:



The EXCLUDE statement parameters are:

MAPNAME= mapname Defines the name of the database map from which items are to be excluded. The mapname must be specified on the DIRECTORY statement, and it cannot be part of a connected directory.

VOCAB=vocabname Defines the name of the vocabulary from which words are to be excluded. If the system vocabulary is being used, VOCAB=SYSVOCAB should be coded.

SEGNAME=segname	Defines the name of a segment whose fields are to be excluded. This parameter may be used as many times as needed. It applies only to maps for IMS (DL/I) databases and VSAM hierarchical data sets.
NAME=name	Defines the name of a field or word to be excluded.

Notes:

- The EXCLUDE statement for the native SQL syntax facility can only exclude the words used by this facility from the vocabulary such as EXTRACT, DEFINE, and so on.
- The SEGNAME or NAME parameters may be used as many times as needed.
- At least one NAME parameter is required when the VOCAB parameter is specified.
- At least one NAME or SEGNAME parameter is required when the MAPNAME parameter is specified.
- The VOCAB and MAPNAME parameters may be used as many times as needed. They may both be specified in the EXCLUDE statement.

  To exclude multiple statements, use multiple NAME statements:

```
EXCLUDE VOCAB=SYSVOCAB,NAME=SORT,NAME=EXTRACT
```

  To exclude statements, fields, and segments, use the following:

```
EXCLUDE VOCAB=SYSVOCAB,NAME=SORT,NAME=EXTRACT,
          MAPNAME=IIDMMAP,NAME=PLANT.ID,SEGNAME=EMP,
          MAPNAME=IIDSMAP,NAME=EMP.NO
```

Figure 4-15 illustrates the directory definition for the PLANT and SKILL test databases.

```
*
 DIRECTORY NAME=IITESTDB,STRAN=IISYSSRT,SLIMIT=10000,
          MAP=IIDMMAP,MAP=IIDSMAP
 FINISH
*
 END
```

Figure 4-15 PLANT and SKILL Test Databases Directory Definition

Defining Terminals

TERM defines the terminals that can be accessed through each directory entry.

TERM Control Statement Group

The TERM control statement group contains the parameters for relating information to the directory about terminals that send and receive messages. The TERMs specified in an inquiry determine which directory is accessed.

Each TERM can only be related to one directory within an application, except in the case of connected directories. This ensures that the user profile defined for that directory is applied to the inquiry. However, multiple TERM statements which point to the same directory can be specified.

TERM Statement

The TERM statement contains the following information:

- CICS terminal ID, user ID, or operator ID as the terminal name
- Directory name to which the TERM is related
- Physical characteristics of the terminal (line width, page height)
- BMS mapset name used by the terminals
- Limits for database calls, VSAM reads, and logical pages
- Terminal mode setting
- Footing map when needed for this terminal
- Whether or not it is a dummy terminal

The format of the TERM statement is:

```
LTERM      NAME={term-name|generic-term-name},
or         DIRECTORY=directory-name,
TERM       MODE=cvs
           (,TERMLTH=number)
           (,TIME=time)
           (,PAGE=num)
           (,BMS=bms-name)
           (,TERMWDTH=width)
           (,ERRHELP=val)
           (,CKPT=ckpt)
           (,DEFINITION=term)
           (,FOOTING=ftg)
           (,APPL=application)
           (,PCPAGE=pcnum)
           (,PCOUT=YES)
```

The TERM statement parameters are:

NAME=term-name

Defines a 4-character terminal ID or an 8-character user ID that is specified in the TYPETERM definition or a 3-character operator ID specified in the CICS segment of a RACF user profile or the Sign-on Table (SNT).

Whether the terminal ID, operator ID, or user ID is used depends on the INTERM parameter for inquiry processing and the OUTTERM parameter for the OUTPUT command in the member CVLCPARM in II.TCVLSRC.

When using VISION:Inquiry in batch mode, terminal names can be up to eight characters.

NAME= generic-term-name

Defines actual characters and wildcard (*) characters. As an example, the generic term name, L*1*, matches any four-character terminal name that starts with L followed by any character, a 1, and finally any character or blank. A generic terminal name can also be all asterisks. When using VISION:Inquiry in batch mode, generic terminal names can be up to eight characters.

DIRECTORY= directory-name

Defines the name of the directory with which this TERM is associated.

MODE=cvs

Defines a character string. Each character is either a 0 or 1 and describes the mode type of the terminal. See below for more information. If this is not specified, MODE=101 is assumed.

c=0

Specifies (non-conversational) continuous mode. This causes VISION:Inquiry to display all the inquiry output without interruption.

c = 1

Specifies conversational mode. VISION:Inquiry writes a checkpoint record where the page or time limit for this terminal is exceeded. This mode parameter only affects page-end checkpointing. Time limit checkpoints occur regardless of the mode specification. This should be used for terminals in the CICS online environment.

v = 0

Is always zero for CICS and batch and is used by the IMS version of VISION:Inquiry only.

s = 0

Perform sorts by a transaction not associated with the terminal. Not implemented in the CICS version.

s = 1

Perform sorts at originating transaction with original terminal attached.

TERMLTH=number

Specifies the number of lines of available output data per page (screen). The CICS version determines this parameter dynamically, but must be supplied when using the OUTPUT parameter to write the results to another terminal (or printer) and when UDO is used.

- If the TERMLTH parameter is not specified, a default of 20 lines assumed.
- VISION:Journey and Intraccess also use this number to determine the page limit checkpoint condition when downloading data.

TIME=time

Specifies the number of database calls and VSAM reads in tens, to be processed before VISION:Inquiry takes a checkpoint. Checkpoints are ignored when running IIBATCH, specifying 'OUTPUT BATCH,' and when using the EXTRACT statement.

If the parameter is omitted, no database call limit is placed on the inquiry (maximum is 3200) unless the SLIMIT parameter has been used on the DIRECTORY statement.

PAGE=num

Specifies the number of logical pages to be processed before VISION:Inquiry takes a checkpoint in conversational processing. Default is 1. The maximum is 255. See the notes below regarding TERM statement parameters.

BMS=bms-name

Defines the name of a predefined BMS mapset name used by VISION:Inquiry to send output. The default name is specified in the member CVLCPARM in II.TCVLSRC; in other words, if not specified, then the default mapset is controlled by the originating transaction. (See Chapter 6, "Programming and Operation Considerations", for details on specifying the default mapset name.)

TERMWDTH=width

Specifies the number of characters that appears in each line for the named terminal. The value should be consistent with the length of the output line defined in the map. When running in batch mode, specify 132. Default is 79.

ERRHELP=val

Specifies whether or not the user receives the ERRHELP text along with an error message. YES is the default. NO causes the add-on ERRHELP text to be suppressed. With this option, the single line error message is all that the user receives.

CKPT=ckpt

Specifies whether a checkpoint is allowed for this terminal; YES is default.

DEFINITION=term

Specifies the name of a previously defined terminal or generic terminal.

The TERM that you are defining takes on the parameters of the previously defined TERM or generic TERM; although, if you enter any parameters other than DEFINITION, the entered parameters override the previously defined parameter. For example, if you enter the following:

```
TERM NAME=DYL11,MODE=111,DEFINITION=DYL10
```

Terminal DYL11 takes on the parameters defined for DYL10, except for the MODE.

FOOTING=ftg

Specifies whether or not the footing map is present in the BMS map. FOOTING=YES causes the footing map FOOTING to be on every page of the output (except last) and FINAL on the last page. Make sure that the mapset contains no footing maps if FOOTING=NO or the dynamic calculation of the lines per page will be incorrect.

Default is FOOTING=NO. (See Chapter 6, "Programming and Operation Considerations".)

APPL=application

Used in conjunction with the DEFINITION parameter. APPL refers back to a TERM definition in a previously defined APPL. You must also use the DIRECTORY parameter so as not to point to a directory in the previously defined APPL.

PCPAGE=pcnum

Specifies the number of logical pages to be extracted before VISION:Inquiry takes a checkpoint in conversational processing. Default is 1; maximum is 255.

This number times the value defined for the TERMLTH parameter gives the number of records extracted before a page limit checkpoint happens for VISION:Journey and Intraccess.

This parameter is only used with the PCE command. See the notes below regarding TERM statement parameters.

PCOUT=YES

Identifies this as a dummy terminal. If the DISPLAY (or FORMAT) statements are used and the output is routed to this terminal using the OUTPUT statement, the output is sent directly to the PC via VISION:Journey.

The characteristics of this dummy terminal are also used when downloading output reports resulting from processing of UDO queries or queries with summary commands to the PC using Intraccess. See the notes below.

Notes:

- VISION:Inquiry can run in two different modes, continuous and conversational:
  - In the continuous (non-conversational mode), VISION:Inquiry processes the inquiry until the call limit is exceeded or to the end if no call limit is set.
  - In the conversational mode, VISION:Inquiry stops processing whenever one of the limit settings is reached. The conversational mode is typically used in the CICS online environment.
- An inquiry that browses through a large database can tie up the CICS region for a long time. To avoid monopolizing the resource, the database administrator can specify two limits that restrict the length of time the transaction stays in the CICS online region. The first limit is a number of database calls/VSAM reads. The second limit is a number of logical pages.
- In conversational mode when CKPT=YES is specified, whenever one of the limits is reached during execution, a VISION:Inquiry checkpoint is taken and the output gathered up to that point is sent to the user. The task is terminated and the terminal is ready for another application. If CKPT=NO is specified, whenever one of the limits is reached during execution, the inquiry is stopped and cannot be continued.
- The terminal operator has the option of continuing from that point on (if CKPT=YES) until another limit is reached or deferring the inquiry to be restarted later. This is a very efficient way of controlling the CICS online region usage.
- In continuous mode, with the exception of batch, the database call/VSAM read limit is respected. However, the inquiry is stopped at that point and cannot be continued. In the batch region, VISION:Inquiry processes the inquiry until all data is retrieved.

- The value specified for TERMLTH parameter multiplied by the PAGE or PCPAGE parameter must not be larger than 32,767.
- Intraccess can only be used in the CICS online environment. The conversational mode and checkpoint facility are applicable. When the checkpoint happens, processing stops and VISION:Inquiry sends the partial output to Intraccess. Then, VISION:Inquiry continues the process for the rest of the query or the next checkpoint.
- The PAGE, BMS, FOOTING, TERMWDTH, and PCOUT parameters have no effect during the download of data (output of Non-UDO queries with no summary commands) using Intraccess.
- Using Intraccess to download a report (output of UDO queries or queries with summary commands), the following rule is in effect.

  The dummy terminal with the name PC and parameter PCOUT=YES must be defined to the system. The parameters of the dummy terminal will be used in the process except:

  - The BMS, FOOTING, and PCPAGE parameters have no effect on the inquiry process.
  - The MODE and CKPT parameters of the terminal specified by Intraccess will be used.

- VISION:Journey can only be used in the online environment. The conversational mode with checkpoint facility is also supported.
- When a terminal uses the PCE command in VISION:Journey, the parameters PAGE, TERMWDTH, BMS, FOOTING, and PCOUT have no effect on the inquiry processing.
- The terminal in the OUTPUT statement when VISION:Journey is used to transfer report records to the PC must be a dummy terminal, defined with parameter PCOUT=YES and with the name PC.
  In this case, all the parameters of the dummy terminal (except the parameters BMS, FOOTING, and PCPAGE, which have no effect on the processing of the inquiry) and the MODE and CKPT parameters of the originating terminal, will be used.
- When you define a dummy terminal for an application, all the directories of that application can use this dummy terminal to transfer report records to the PC, no matter what is specified in the DIRECTORY parameter of the TERM statement for the dummy terminal.

Defining the Test Data

Input to IIGEN is usually created in an input data set. Figure 4-16 displays part of the source library member of II.TCVLSRC (IIDMGEN) which can be used as input to IIGEN.

IIGEN Sample Input - II.TCVLSRC (IIDMGEN)

The following examples use the test data distributed on the installation tape:

- The IMS (DL/I) test databases (PLANT and SKILL),
- The VSAM test data sets (VSPLANT, VSSKILL, VSHPLANT and VSHSKILL), and
- The DB2 test tables and views.

It also shows the examples of the DB2 tables sent by IBM with the DB2 licensed program. If you are unfamiliar with the test data, see Chapter 3, "System Components".

```
APPL NAME=IQIO
*
*        MAPGEN FOR PLANT SAMPLE DATABASE
*
 MAPGEN  MAP=IIDMMAP,DBD=IIDBDDM,NAME=PLANT,
         DESC="TEST DATABASE - PRODUCTS AND EMPLOYEES"
 SEGMENT SEGM=PLANT,PARENT=0,BYTES=40,KEY=PLANTKEY
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=PLANT.ID,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=6,LENGTH=2,TYPE=C,NAME=PLANT.REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   START=8,LENGTH=7,TYPE=C,NAME=PLANT.PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   START=15,LENGTH=25,TYPE=C,NAME=PLANT.NAME,
         DESC="NAME OF PLANT"
 SEGMENT SEGM=PROD,PARENT=PLANT,BYTES=35,KEY=PRODKEY
 FIELD   START=1,LENGTH=2,TYPE=C,NAME=PROD.CODE,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR PRODUCT"
 FIELD   START=3,LENGTH=4,TYPE=B,NAME=PROD.QTY,
         DESC="INVENTORY ON HAND"
 FIELD   START=7,LENGTH=4,TYPE=B,NAME=PROD.AMT,SCALE=+2,
         DESC="SELLING PRICE"
 FIELD   START=11,LENGTH=25,TYPE=C,NAME=PROD.DESC,
         DESC="NAME OF PRODUCT"
 SEGMENT SEGM=EMP,PARENT=PLANT,BYTES=31,KEY=EMPKEY
 FIELD   START=1,LENGTH=5,TYPE=N,NAME=EMP.NO,KEY=SEQ-U,OUTEDIT=NONE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   START=6,LENGTH=1,TYPE=C,NAME=EMP.SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   START=7,LENGTH=25,TYPE=C,NAME=EMP.NAME,
         DESC="NAME OF EMPLOYEE"
 SEGMENT SEGM=SAL,PARENT=EMP,BYTES=11,KEY=SALKEY
 FIELD   START=1,LENGTH=2,TYPE=N,NAME=SAL.YEAR,KEY=SEQ-U,
         DESC="CALENDAR YEAR"
 FIELD   START=3,LENGTH=5,TYPE=P,NAME=SAL.YTD,SCALE=+2,
         DESC="TOTAL SALARY"
 FIELD   START=8,LENGTH=4,TYPE=P,NAME=SAL.DED,SCALE=+2,
         DESC="SALARY DEDUCTIONS"
 FIELD   START=8,LENGTH=4,TYPE=Y,SUBSTR=(1,5),NAME=SAL.DED.T,
```

Figure 4-16 Input to IIGEN (Page 1 of 8)

```
         DESC="SALARY DEDUCTIONS (WHOLE DOLLARS)"
 FIELD   START=8,LENGTH=4,TYPE=Y,SUBSTR=(6,2),NAME=SAL.DED.DEC,
         DESC="SALARY DEDUCTIONS (ODD CENTS)"
 SEGMENT SEGM=ED,PARENT=EMP,BYTES=14,KEY=EDKEY
 FIELD   START=1,LENGTH=2,TYPE=N,NAME=ED.YEAR,KEY=SEQ-U,
         DESC="YEAR OF GRADUATION"
 FIELD   START=3,LENGTH=2,TYPE=C,NAME=ED.DEGREE,
         DESC="DEGREE ATTAINED"
 FIELD   START=5,LENGTH=10,TYPE=C,NAME=ED.SCHOOL,
         DESC="SCHOOL ATTENDED"
 SEGMENT SEGM=SUB,BYTES=12,PARENT=ED,KEY=SUBKEY
 FIELD   START=1,LENGTH=2,TYPE=C,NAME=SUB.GRADE,
         DESC="OVERALL GRADE WITHIN MAJOR FIELD OF STUDY"
 FIELD   START=3,LENGTH=10,TYPE=C,NAME=SUB.NAME,KEY=SEQ-U,
         DESC="MAJOR FIELD OF STUDY"
 FINISH
*
*        MAPGEN FOR SKILL SAMPLE DATABASE
*
 MAPGEN  MAP=IIDSMAP,DBD=IIDBDDS,NAME=SKILL,
         DESC="TEST DATABASE - JOB CLASSIFICATIONS"
 SEGMENT SEGM=SKILL,PARENT=0,BYTES=23,KEY=SKILLKEY
 FIELD   START=1,LENGTH=2,TYPE=P,NAME=SKILL.CODE,NAME=SC,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   START=3,LENGTH=20,TYPE=C,NAME=SKILL.NAME,NAME=SN,
         DESC="TITLE OF JOB SKILL"
 SEGMENT SEGM=PLANT,PARENT=SKILL,BYTES=4,KEY=PLANTKEY
 FIELD   START=1,LENGTH=4,TYPE=P,NAME=PLANT.ID,NAME=PID,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR PLANT"
 SEGMENT SEGM=EMP,PARENT=PLANT,BYTES=5,KEY=EMPKEY
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=EMP.NO,NAME=EN,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FINISH
*
*        MAPGEN FOR VSPLANT SAMPLE VSAM FILE
*
 MAPGEN  MAP=VSPLMAP,FILE=VSPLDS,NAME=VSPLANT,DBTYPE=VSAMKSDS,
         DESC="VSAM KSDS TEST FILE - EMPLOYEES"
 RECORD  BYTES=80
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=VSPLANT.ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=6,LENGTH=5,TYPE=N,NAME=VSEMP.NO,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   START=11,LENGTH=25,TYPE=C,NAME=VSEMP.NAME,
         DESC="NAME OF EMPLOYEE"
 FIELD   START=36,LENGTH=1,TYPE=C,NAME=VSEMP.SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   START=37,LENGTH=2,TYPE=C,NAME=VSED.DEGREE,
         DESC="DEGREE ATTAINED"
 FIELD   START=39,LENGTH=7,TYPE=N,NAME=VSSAL.YTD,SCALE=+2,
         DESC="TOTAL SALARY"
 FIELD   START=46,LENGTH=7,TYPE=N,NAME=VSSAL.DED,SCALE=+2,
         DESC="SALARY DEDUCTIONS"
 FINISH
*
*        MAPGEN FOR VSSKILL SAMPLE VSAM FILE
*
 MAPGEN  MAP=VSSKMAP,FILE=VSSKDS,NAME=VSSKILL,DBTYPE=VSAMRRDS,
         DESC="VSAM RRDS TEST FILE - JOB SKILLS"
 RECORD  BYTES=80
 FIELD   START=1,LENGTH=5,TYPE=N,NAME=VSPLANT.ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=6,LENGTH=5,TYPE=C,NAME=VSEMP.NO,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   START=11,LENGTH=2,TYPE=N,NAME=VSSKILL.CODE,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   START=13,LENGTH=20,TYPE=C,NAME=VSSKILL.NAME,
         DESC="TITLE OF JOB SKILL"
 FINISH
*
*        MAPGEN FOR PLANT SAMPLE VSAM HIERARCHICAL FILE
*
 MAPGEN  MAP=VSHPLMAP,FILE=VSHPLDS,NAME=VSHPLANT,DBTYPE=VSMHKSDS,
```

Figure 4-16 Input to IIGEN (Page 2 of 8)

```
         DESC="VSMH TEST FILE - PRODUCTS AND EMPLOYEES"
*
 RECORD  BYTES=747,TYPE=V
*
 SEGMENT SEGM=VSHPLANT,PARENT=0,BYTES=41
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=VSHPLANT.ID,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=6,LENGTH=2,TYPE=C,NAME=VSHPLANT.REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   START=8,LENGTH=7,TYPE=C,NAME=VSHPLANT.PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   START=15,LENGTH=25,TYPE=C,NAME=VSHPLANT.NAME,
         DESC="NAME OF PLANT"
 FIELD   START=40,LENGTH=2,TYPE=B,NAME=PLEMPOCR,NAME=VSHPLANT.EMPOCR,
         OCRCNT=YES,DESC="NO. OF EMPLOYEE SEGMENTS PER PLANT"
*
 SEGMENT SEGM=VSHPROD,PARENT=VSHPLANT,BYTES=35,OCRNUM=4
 FIELD   START=1,LENGTH=2,TYPE=C,NAME=VSHPROD.CODE,
         DESC="IDENTIFICATION CODE FOR PRODUCT"
 FIELD   START=3,LENGTH=4,TYPE=B,NAME=VSHPROD.QTY,
         DESC="INVENTORY ON HAND"
 FIELD   START=7,LENGTH=4,TYPE=B,NAME=VSHPROD.AMT,SCALE=+2,
         DESC="SELLING PRICE"
 FIELD   START=11,LENGTH=25,TYPE=C,NAME=VSHPROD.DESC,
         DESC="NAME OF PRODUCT"
*
 SEGMENT SEGM=VSHEMP,PARENT=VSHPLANT,BYTES=35,OCRFLD=PLEMPOCR
 FIELD   START=1,LENGTH=5,TYPE=N,NAME=VSHEMP.NO,OUTEDIT=NONE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   START=6,LENGTH=1,TYPE=C,NAME=VSHEMP.SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   START=7,LENGTH=25,TYPE=C,NAME=VSHEMP.NAME,
         DESC="NAME OF EMPLOYEE"
 FIELD   START=32,LENGTH=2,TYPE=B,NAME=EMSALOCR,NAME=VSHEMP.SALOCR,
         OCRCNT=YES,DESC="NO. OF SALARY SEGMENTS PER EMP"
 FIELD   START=34,LENGTH=2,TYPE=B,NAME=EMPEDOCR,NAME=VSHEMP.EDOCR,
         OCRCNT=YES,DESC="NO. OF EDUCATION SEGMENTS PER EMP"
*
 SEGMENT SEGM=VSHSAL,PARENT=VSHEMP,BYTES=11,OCRFLD=EMSALOCR
 FIELD   START=1,LENGTH=2,TYPE=N,NAME=VSHSAL.YEAR,
         DESC="CALENDAR YEAR"
 FIELD   START=3,LENGTH=5,TYPE=P,NAME=VSHSAL.YTD,SCALE=+2,
         DESC="TOTAL SALARY"
 FIELD   START=8,LENGTH=4,TYPE=P,NAME=VSHSAL.DED,SCALE=+2,
         DESC="SALARY DEDUCTIONS"
 FIELD   START=8,LENGTH=4,TYPE=Y,SUBSTR=(1,5),NAME=VSHSAL.DED.T,
         DESC="SALARY DEDUCTIONS (WHOLE DOLLARS)"
 FIELD   START=8,LENGTH=4,TYPE=Y,SUBSTR=(6,2),NAME=VSHSAL.DED.DEC,
         DESC="SALARY DEDUCTIONS (ODD CENTS)"
*
 SEGMENT SEGM=VSHED,PARENT=VSHEMP,BYTES=16,OCRFLD=EMPEDOCR
 FIELD   START=1,LENGTH=2,TYPE=N,NAME=VSHED.YEAR,
         DESC="YEAR OF GRADUATION"
 FIELD   START=3,LENGTH=2,TYPE=C,NAME=VSHED.DEGREE,
         DESC="DEGREE ATTAINED"
 FIELD   START=5,LENGTH=10,TYPE=C,NAME=VSHED.SCHOOL,
         DESC="SCHOOL ATTENDED"
 FIELD   START=15,LENGTH=2,TYPE=P,NAME=EDSUBOCR,NAME=VSHED.SUBOCR,
         OCRCNT=YES,DESC="NO. OF SUBJECT SEGMENTS PER EDUCATION"
*
 SEGMENT SEGM=VSHSUB,BYTES=12,PARENT=VSHED,OCRFLD=EDSUBOCR
 FIELD   START=1,LENGTH=2,TYPE=C,NAME=VSHSUB.GRADE,
         DESC="OVERALL GRADE WITHIN MAJOR FIELD OF STUDY"
 FIELD   START=3,LENGTH=10,TYPE=C,NAME=VSHSUB.NAME,
         DESC="MAJOR FIELD OF STUDY"
 FINISH
*
*        MAPGEN FOR SKILL SAMPLE VSAM HIERARCHICAL FILE
*
 MAPGEN  MAP=VSHSKMAP,FILE=VSHSKDS,NAME=VSHSKILL,DBTYPE=VSMHKSDS,
         DESC="VSMH TEST FILE - JOB CLASSIFICATIONS"
*
```

Figure 4-16 Input to IIGEN (Page 3 of 8)

```
 RECORD  BYTES=90,TYPE=V
*
 SEGMENT SEGM=VSHSKILL,PARENT=0,BYTES=24
 FIELD   START=1,LENGTH=2,TYPE=P,NAME=VSHSKILL.CODE,KEY=SEQ-U,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   START=3,LENGTH=20,TYPE=C,NAME=VSHSKILL.NAME,
         DESC="TITLE OF JOB SKILL"
 FIELD   START=23,LENGTH=2,TYPE=P,NAME=SKPLTOCR,NAME=VSHSKILL.PLNTOCR,
         OCRCNT=YES,DESC="NO. OF PLANT SEGMENTS PER SKILL"
*
 SEGMENT SEGM=VSHPLANT,PARENT=VSHSKILL,BYTES=6,OCRFLD=SKPLTOCR
 FIELD   START=1,LENGTH=4,TYPE=P,NAME=VSHPLANT.ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   START=5,LENGTH=2,TYPE=B,NAME=PLEMPOCR,NAME=VSHPLANT.EMPOCR,
         OCRCNT=YES,DESC="NO. OF EMPLOYEE SEGMENTS PER PLANT"
*
 SEGMENT SEGM=VSHEMP,PARENT=VSHPLANT,BYTES=5,OCRFLD=PLEMPOCR
 FIELD   START=1,LENGTH=5,TYPE=C,NAME=VSHEMP.NO,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FINISH
*
*        MAPGEN FOR DB2 - PLANT/SKILL-LIKE TABLES
*
 MAPGEN  MAP=IIPL1,NAME=PLANT2,AUTHID=DYLINQ,
         TABLENAME=IIPLANT,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - PLANTS"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT.ID,DBNAME=PLANT,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=2,TYPE=C,NAME=REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   LENGTH=7,TYPE=C,NAME=PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   LENGTH=25,TYPE=C,NAME=NAME,
         DESC="NAME OF PLANT"
 FINISH
*
 MAPGEN  MAP=IIPL2,NAME=PL2PROD,AUTHID=DYLINQ,
         TABLENAME=IIPLANT_PROD,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - PLANTS AND PRODUCTS"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT.ID,DBNAME=PLANT,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=2,TYPE=C,NAME=CODE,
         DESC="IDENTIFICATION CODE FOR PRODUCT"
 FIELD   LENGTH=4,TYPE=B,NAME=QTY,
         DESC="INVENTORY ON HAND"
 FIELD   LENGTH=8,TYPE=P,NAME=AMT,SCALE=+2,
         DESC="SELLING PRICE"
 FIELD   LENGTH=25,TYPE=C,NAME=DESC,
         DESC="NAME OF PRODUCT"
 FINISH
*
 MAPGEN  MAP=IIPL3,NAME=PL2EMP,AUTHID=DYLINQ,
         TABLENAME=IIPLANT_EMP,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - PLANTS AND EMPLOYEES"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT.ID,DBNAME=PLANT,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP.NO,DBNAME=EMPLOYEE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   LENGTH=1,TYPE=C,NAME=SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   LENGTH=25,TYPE=C,NAME=NAME,
         DESC="NAME OF EMPLOYEE"
 FINISH
*
 MAPGEN  MAP=IIPL4,NAME=EMP2SAL,AUTHID=DYLINQ,
         TABLENAME=IIEMP_SAL,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - EMPLOYEES AND SALARIES"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP.NO,DBNAME=EMPLOYEE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
```

Figure 4-16 Input to IIGEN (Page 4 of 8)

```
 FIELD   LENGTH=2,TYPE=C,NAME=YEAR,
         DESC="CALENDAR YEAR"
 FIELD   LENGTH=5,TYPE=P,NAME=YTD,DBNAME=YEAR_TO_DATE,SCALE=+2,
         DESC="TOTAL SALARY"
 FIELD   LENGTH=4,TYPE=P,NAME=DED,DBNAME=DEDUCTIONS,SCALE=+2,
         DESC="SALARY DEDUCTIONS"
 FINISH
*
 MAPGEN  MAP=IIPL5,NAME=EMP2ED,AUTHID=DYLINQ,
         TABLENAME=IIEMP_ED,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - EMPLOYEES AND EDUCATION"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP.NO,DBNAME=EMPLOYEE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   LENGTH=2,TYPE=C,NAME=YEAR,
         DESC="YEAR OF GRADUATION"
 FIELD   LENGTH=2,TYPE=C,NAME=DEGREE,
         DESC="DEGREE ATTAINED"
 FIELD   LENGTH=10,TYPE=C,NAME=SCHOOL,
         DESC="SCHOOL ATTENDED"
 FINISH
*
 MAPGEN  MAP=IIPL6,NAME=ED2SUB,AUTHID=DYLINQ,
         TABLENAME=IIEMP_ED_SUB,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - SUBJECTS OF EDUCATION"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP.NO,DBNAME=EMPLOYEE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   LENGTH=2,TYPE=C,NAME=YEAR,
         DESC="YEAR OF GRADUATION"
 FIELD   LENGTH=2,TYPE=C,NAME=GRADE,
         DESC="OVERALL GRADE WITHIN MAJOR FIELD OF STUDY"
 FIELD   LENGTH=10,TYPE=C,NAME=NAME,
         DESC="MAJOR FIELD OF STUDY"
 FINISH
*
 MAPGEN  MAP=IISK1,NAME=SKILL2,AUTHID=DYLINQ,
         TABLENAME=IISKILL,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - JOB CLASSIFICATIONS"
 FIELD   LENGTH=2,TYPE=P,NAME=SKILL.CODE,DBNAME=SKILL,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   LENGTH=20,TYPE=C,NAME=NAME,
         DESC="TITLE OF JOB SKILL"
 FINISH
*
 MAPGEN  MAP=IISK2,NAME=SK2PLANT,AUTHID=DYLINQ,
         TABLENAME=IISKILL_PLANT,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - SKILLS AND PLANTS"
 FIELD   LENGTH=2,TYPE=P,NAME=SKILL.CODE,DBNAME=SKILL,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT.ID,DBNAME=PLANT,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FINISH
*
 MAPGEN  MAP=IISK3,NAME=SK2EMP,AUTHID=DYLINQ,
         TABLENAME=IISKILL_EMP,
         DBTYPE=DB2,
         DESC="TEST DB2 TABLE - JOB SKILLS & EMPLOYEES"
 FIELD   LENGTH=2,TYPE=P,NAME=SKILL.CODE,DBNAME=SKILL,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP.NO,DBNAME=EMPLOYEE,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FINISH
*
 MAPGEN  MAP=IIVW1,NAME=PRODUCTS,AUTHID=DYLINQ,
         TABLENAME=IIPRODUCTS,
         DBTYPE=DB2,
         DESC="TEST VIEW - PLANT AND PRODUCT DATA"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT_ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
```

Figure 4-16 Input to IIGEN (Page 5 of 8)

```
 FIELD   LENGTH=25,TYPE=C,NAME=PLANT_NAME,
         DESC="NAME OF PLANT"
 FIELD   LENGTH=7,TYPE=C,NAME=PLANT_PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   LENGTH=2,TYPE=C,NAME=PLANT_REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   LENGTH=2,TYPE=C,NAME=PROD_CODE,
         DESC="IDENTIFICATION CODE FOR PRODUCT"
 FIELD   LENGTH=25,TYPE=C,NAME=PROD_DESC,
         DESC="NAME OF PRODUCT"
 FIELD   LENGTH=8,TYPE=P,NAME=PROD_AMT,SCALE=+2,
         DESC="SELLING PRICE"
 FIELD   LENGTH=4,TYPE=B,NAME=PROD_QTY,
         DESC="INVENTORY ON HAND"
 FINISH
*
 MAPGEN  MAP=IIVW2,NAME=SALARIES,AUTHID=DYLINQ,
         TABLENAME=IISALARIES,
         DBTYPE=DB2,
         DESC="TEST VIEW - PLANT, EMPLOYEE AND SALARY DATA"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT_ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=25,TYPE=C,NAME=PLANT_NAME,
         DESC="NAME OF PLANT"
 FIELD   LENGTH=7,TYPE=C,NAME=PLANT_PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   LENGTH=2,TYPE=C,NAME=PLANT_REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP_NO,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   LENGTH=25,TYPE=C,NAME=EMP_NAME,
         DESC="NAME OF EMPLOYEE"
 FIELD   LENGTH=1,TYPE=C,NAME=EMP_SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   LENGTH=2,TYPE=C,NAME=SAL_YEAR,
         DESC="CALENDAR YEAR"
 FIELD   LENGTH=5,TYPE=P,NAME=SAL_YTD,SCALE=+2,
         DESC="TOTAL SALARY"
 FIELD   LENGTH=4,TYPE=P,NAME=SAL_DED,SCALE=+2,
         DESC="SALARY DEDUCTIONS"
 FINISH
*
 MAPGEN  MAP=IIVW3,NAME=SUBJECTS,AUTHID=DYLINQ,
         TABLENAME=IIEDUCATION,
         DBTYPE=DB2,
         DESC="TEST VIEW - EMPLOYEE EDUCATION DATA"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT_ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=25,TYPE=C,NAME=PLANT_NAME,
         DESC="NAME OF PLANT"
 FIELD   LENGTH=7,TYPE=C,NAME=PLANT_PHONE,
         DESC="MAIN SWITCHBOARD PHONE NUMBER"
 FIELD   LENGTH=2,TYPE=C,NAME=PLANT_REGION,
         DESC="GEOGRAPHIC AREA"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP_NO,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FIELD   LENGTH=25,TYPE=C,NAME=EMP_NAME,
         DESC="NAME OF EMPLOYEE"
 FIELD   LENGTH=1,TYPE=C,NAME=EMP_SEX,
         DESC="EMPLOYEE'S SEX"
 FIELD   LENGTH=2,TYPE=C,NAME=ED_YEAR,
         DESC="YEAR OF GRADUATION"
 FIELD   LENGTH=2,TYPE=C,NAME=ED_DEGREE,
         DESC="DEGREE ATTAINED"
 FIELD   LENGTH=10,TYPE=C,NAME=ED_SCHOOL,
         DESC="SCHOOL ATTENDED"
 FIELD   LENGTH=10,TYPE=C,NAME=SUB_NAME,
         DESC="MAJOR FIELD OF STUDY"
 FIELD   LENGTH=2,TYPE=C,NAME=SUB_GRADE,
         DESC="OVERALL GRADE WITHIN MAJOR FIELD OF STUDY"
 FINISH
*
 MAPGEN  MAP=IIVW4,NAME=SKILLS,AUTHID=DYLINQ,
```

Figure 4-16 Input to IIGEN (Page 6 of 8)

```
         TABLENAME=IISKILLS,
         DBTYPE=DB2,
         DESC="TEST VIEW - JOB CLASSIFICATIONS"
 FIELD   LENGTH=2,TYPE=P,NAME=SKILL_CODE,
         DESC="IDENTIFICATION CODE FOR JOB SKILL"
 FIELD   LENGTH=20,TYPE=C,NAME=SKILL_NAME,
         DESC="TITLE OF JOB SKILL"
 FIELD   LENGTH=5,TYPE=C,NAME=PLANT_ID,
         DESC="IDENTIFICATION CODE FOR PLANT"
 FIELD   LENGTH=5,TYPE=C,NAME=EMP_NO,
         DESC="IDENTIFICATION CODE FOR EMPLOYEE"
 FINISH
*
*        MAPGEN FOR DB2 - IBM-SUPPLIED SAMPLE TABLES
*
 MAPGEN  MAP=IBEMPL,NAME=EMPL,AUTHID=DSN82,TABLENAME=TEMPL,DBTYPE=DB2
 FIELD   NAME=EMPNO,TYPE=C,LENGTH=6
 FIELD   NAME=FIRSTNME,TYPE=C,LENGTH=12
 FIELD   NAME=MIDINIT,TYPE=C,LENGTH=1
 FIELD   NAME=LASTNAME,TYPE=C,LENGTH=15
 FIELD   NAME=WORKDEPT,TYPE=C,LENGTH=3
 FIELD   NAME=PHONENO,TYPE=C,LENGTH=4
 FIELD   NAME=HIREDATE,TYPE=P,LENGTH=4
 FIELD   NAME=JOBCODE,TYPE=P,LENGTH=2
 FIELD   NAME=EDUCLVL,TYPE=B,LENGTH=2
 FIELD   NAME=SEX,TYPE=C,LENGTH=1
 FIELD   NAME=BRTHDATE,TYPE=P,LENGTH=4
 FIELD   NAME=SALARY,TYPE=P,LENGTH=5,SCALE=+2
 FINISH
*
 MAPGEN  MAP=IBDEPT,NAME=DEPT,AUTHID=DSN82,TABLENAME=TDEPT,DBTYPE=DB2
 FIELD   NAME=DEPTNO,TYPE=C,LENGTH=3
 FIELD   NAME=DEPTNAME,TYPE=C,LENGTH=36
 FIELD   NAME=MGRNO,TYPE=C,LENGTH=6
 FIELD   NAME=ADMRDEPT,TYPE=C,LENGTH=3
 FINISH
*
 MAPGEN  MAP=IBPROJ,NAME=PROJ,AUTHID=DSN82,TABLENAME=TPROJ,DBTYPE=DB2
 FIELD   NAME=PROJNO,TYPE=C,LENGTH=6
 FIELD   NAME=PROJNAME,TYPE=C,LENGTH=24
 FIELD   NAME=DEPTNO,TYPE=C,LENGTH=3
 FIELD   NAME=RESPEMP,TYPE=C,LENGTH=6
 FIELD   NAME=PRSTAFF,TYPE=P,LENGTH=3,SCALE=+2
 FIELD   NAME=PRSTDATE,TYPE=P,LENGTH=4
 FIELD   NAME=PRENDATE,TYPE=P,LENGTH=4
 FIELD   NAME=MAJPROJ,TYPE=C,LENGTH=6
 FINISH
*
 MAPGEN  MAP=IBACTYPE,NAME=ACTYPE,AUTHID=DSN82,TABLENAME=TACTYPE,
         DBTYPE=DB2
 FIELD   NAME=ACTNO,TYPE=B,LENGTH=2
 FIELD   NAME=ACTKWD,TYPE=C,LENGTH=6
 FIELD   NAME=ACTDESC,TYPE=C,LENGTH=20
 FINISH
*
 MAPGEN  MAP=IBPROJAC,NAME=PROJAC,AUTHID=DSN82,TABLENAME=TPROJAC,
         DBTYPE=DB2
 FIELD   NAME=PROJNO,TYPE=C,LENGTH=6
 FIELD   NAME=ACTNO,TYPE=B,LENGTH=2
 FIELD   NAME=ACSTAFF,TYPE=P,LENGTH=3,SCALE=+2
 FIELD   NAME=ACSTDATE,TYPE=P,LENGTH=4
 FIELD   NAME=ACENDATE,TYPE=P,LENGTH=4
 FINISH
*
 MAPGEN  MAP=IBEMPRAC,NAME=EMPRAC,AUTHID=DSN82,TABLENAME=TEMPRAC,
         DBTYPE=DB2
 FIELD   NAME=EMPNO,TYPE=C,LENGTH=6
 FIELD   NAME=PROJNO,TYPE=C,LENGTH=6
 FIELD   NAME=ACTNO,TYPE=B,LENGTH=2
 FIELD   NAME=EMPTIME,TYPE=P,LENGTH=3,SCALE=+2
 FIELD   NAME=EMSTDATE,TYPE=P,LENGTH=4
 FIELD   NAME=EMENDATE,TYPE=P,LENGTH=4
```

Figure 4-16 Input to IIGEN (Page 7 of 8)

```
 FINISH
*
*        DIRECTORY FOR
*        PLANT AND SKILL SAMPLE DATABASES,
*        VSPLANT AND VSSKILL VSAM FILES,
*        VSHPLANT AND VSHSKILL VSAM HIERARCHICAL FILES,
*        PLANT- AND SKILL-LIKE DB2 TABLES,
*        AND IBM-SUPPLIED DB2 TEST TABLES
*
 DIRECTORY NAME=IIDMDIR,
* IMS DATABASES
                     MAP=IIDMMAP,MAP=IIDSMAP,
* VSAM FILES
                     MAP=VSPLMAP,MAP=VSSKMAP,
* VSAM HIERARCHICAL FILES
                     MAP=VSHPLMAP,MAP=VSHSKMAP,
* DB2 TABLES & VIEWS
                     MAP=IIPL1,MAP=IIPL2,
                     MAP=IIPL3,MAP=IIPL4,MAP=IIPL5,
                     MAP=IIPL6,MAP=IISK1,MAP=IISK2,MAP=IISK3,
                     MAP=IIVW1,MAP=IIVW2,MAP=IIVW3,MAP=IIVW4,
* IBM DB2 TABLES
                     MAP=IBEMPL,MAP=IBDEPT,MAP=IBPROJ,MAP=IBACTYPE,
                     MAP=IBPROJAC,MAP=IBEMPRAC,
*
                     SLIMIT=10000,
                     VOCAB=SYSVOCAB
 FINISH
*
*        TERMINAL DEFINITIONS
*
 TERM   NAME=BATCH,DIRECTORY=IIDMDIR,TERMWDTH=132,TERMLTH=55,MODE=000
 FINISH
 TERM   NAME=********,DIRECTORY=IIDMDIR,TERMWDTH=79,TERMLTH=20,
        MODE=101,TIME=10,PAGE=2
 FINISH
 TERM   NAME=UDO,DIRECTORY=IIDMDIR,TERMWDTH=79,TERMLTH=12,MODE=101,
        TIME=10,PAGE=2,BMS=INQUDO
 FINISH
 END
```

Figure 4-16 Input to IIGEN (Page 8 of 8)

The IIGEN input, II.TCVLSRC (IIDMGEN), also contains the MAPGEN for the VISION:Journey download data set and the TERM statement for a dummy terminal, but Figure 4-16 does not display the MAPGEN and TERM statements used by VISION:Journey and Intraccess.

IIGEN Sample Output

The output listing produced by IIGEN consists of an audit listing of the input stream and a listing of diagnostic messages. Figure 4-17 depicts a typical application generation output messages listing.

```
12/26/2002 VISION:Inquiry 6.5 FOR CICS SYSTEM DEFINITION AND GENERATION DIAGNOSTICS

MESSAGE  AT  LVL          MESSAGE

NUMBER  STMT

IXG0084   30 00  DATABASE/VSAM MAP 'IIDMMAP ' HAS BEEN GENERATED.
IXG0084   39 00  DATABASE/VSAM MAP 'IIDSMAP ' HAS BEEN GENERATED.
IXG0084   49 00  DATABASE/VSAM MAP 'VSPLMAP ' HAS BEEN GENERATED.
IXG0084   56 00  DATABASE/VSAM MAP 'VSSKMAP ' HAS BEEN GENERATED.
IXG0084   90 00  DATABASE/VSAM MAP 'VSHPLMAP' HAS BEEN GENERATED.
IXG0084  102 00  DATABASE/VSAM MAP 'VSHSKMAP' HAS BEEN GENERATED.
IXG0084  108 00  DATABASE/VSAM MAP 'IIPL1   ' HAS BEEN GENERATED.
IXG0084  115 00  DATABASE/VSAM MAP 'IIPL2   ' HAS BEEN GENERATED.
IXG0084  121 00  DATABASE/VSAM MAP 'IIPL3   ' HAS BEEN GENERATED.
IXG0084  127 00  DATABASE/VSAM MAP 'IIPL4   ' HAS BEEN GENERATED.
IXG0084  133 00  DATABASE/VSAM MAP 'IIPL5   ' HAS BEEN GENERATED.
IXG0084  139 00  DATABASE/VSAM MAP 'IIPL6   ' HAS BEEN GENERATED.
IXG0084  143 00  DATABASE/VSAM MAP 'IISK1   ' HAS BEEN GENERATED.
IXG0084  147 00  DATABASE/VSAM MAP 'IISK2   ' HAS BEEN GENERATED.
IXG0084  151 00  DATABASE/VSAM MAP 'IISK3   ' HAS BEEN GENERATED.
IXG0084  161 00  DATABASE/VSAM MAP 'IIVW1   ' HAS BEEN GENERATED.
IXG0084  173 00  DATABASE/VSAM MAP 'IIVW2   ' HAS BEEN GENERATED.
IXG0084  187 00  DATABASE/VSAM MAP 'IIVW3   ' HAS BEEN GENERATED.
IXG0084  193 00  DATABASE/VSAM MAP 'IIVW4   ' HAS BEEN GENERATED.
IXG0084  207 00  DATABASE/VSAM MAP 'IBEMPL  ' HAS BEEN GENERATED.
IXG0084  213 00  DATABASE/VSAM MAP 'IBDEPT  ' HAS BEEN GENERATED.
IXG0084  223 00  DATABASE/VSAM MAP 'IBPROJ  ' HAS BEEN GENERATED.
IXG0084  228 00  DATABASE/VSAM MAP 'IBACTYPE' HAS BEEN GENERATED.
IXG0084  235 00  DATABASE/VSAM MAP 'IBPROJAC' HAS BEEN GENERATED.
IXG0084  243 00  DATABASE/VSAM MAP 'IBEMPRAC' HAS BEEN GENERATED.
IXG0084  245 00  DIRECTORY 'IIDMDIR' HAS BEEN GENERATED.
IXG0057  246 00  A DIRECTORY LTERM/TERM NAME 'BATCH   ' HAS BEEN INSERTED FOR
IXG0058  246 00  THE DIRECTORY NAMED 'IIDMDIR '
IXG0057  248 00  A DIRECTORY LTERM/TERM NAME '********' HAS BEEN INSERTED FOR
IXG0058  248 00  THE DIRECTORY NAMED 'IIDMDIR '
IXG0057  250 00  A DIRECTORY LTERM/TERM NAME 'UDO     ' HAS BEEN INSERTED FOR
IXG0058  250 00  THE DIRECTORY NAMED 'IIDMDIR '

IXG9228      00  34 DIAGNOSTIC MESSAGES PRINTED.
IXG9229      00  HIGHEST SEVERITY LEVEL ENCOUNTERED = 0.
```

Figure 4-17 Output Messages from IIGEN

IIGEN Messages

As the input stream is processed, input statements are numbered. All diagnostic messages appear separately from the input statements. These messages inform the user of any errors encountered during processing and of the action taken. Each message is printed with an accompanying severity level that reflects error conditions or actions taken.

The following severity levels may appear with a diagnostic message:

0	Information message that documents a successful action.
4	Warning message indicating a potential error.
8	Error message indicating an error condition occurred during statement processing. The action specified for the statement is not taken.
12	Error message indicating that a serious error has occurred. Further processing is not taken for this application.
16	Terminating error

The highest severity level encountered is returned as a completion code to the operating system.

If the highest severity level is 12 or greater, the system database may be damaged or out of space. It may have to be re-created.

Maintaining an Application

The regeneration of a MAPGEN, DIRECTORY, or VOCABULARY within an existing application (APPL) causes an automatic deletion and replacement of the affected entity. Reorganization of space within the system database is automatic with this action.

The directories and terminals within an existing application can be updated using the UPDATEDIR and TERM statements, respectively.

The system database is designed to maximize control and security. The system database entries that control the processing of inquiries are reconstructed each time the system database is created or maintained. This means that individual updates to the system database must be controlled through a central point. This minimizes the exposure of installations to individual users changing definitions or profiles.

Additionally, due to the system database structure, the following rules must be adhered to whenever updating any element of the system database:

1. MAPGEN
   If any statement in the MAPGEN is modified, the entire MAPGEN must be re-entered, and directories including that MAPGEN must be updated or re-entered in the system database.
2. DIRECTORY
   Whenever a directory needs to be modified, you can use:
   - The UPDATEDIR statement to update the vocabulary, the map, and the directory parameters.
   - The DIRECTORY statement to replace the existing directory.

   In either case, only that item must be re-entered in the system database. All other entries in the system database remain intact.
3. TERM
   If a TERM needs to be modified, only that TERM needs to be re-entered in the system database. All other entries in the system database need not be re-entered. The parameters specified in the new TERM definition are updated and the others remain the same as the existing parameters.

UPDATEDIR Statement

Use the UPDATEDIR statement of IIGEN to update the existing directories.

The format of the UPDATEDIR statement is:

```
UPDATEDIR   NAME=name(,VOCAB=vocabname)(,MAP=mapname)
            (,CTRAN=dirtran,CDIR=dirname)
            (,SLIMIT=num)
```

The UPDATEDIR statement parameters are:

NAME=name
Represents the name of an existing directory in the system database.

VOCAB=vocabname
Represents the name of a user vocabulary that was previously defined using the SYSTEM and NAME control statements, which are explained in Chapter 5, "The Utilities".

- When this parameter is specified, it replaces the vocabulary of the existing directory.
- If the parameter is not present, the vocabulary of the directory will not be replaced.

MAP=mapname
Represents the name of the map to be updated or added to the directory. The map must be previously created in the system database using the MAPGEN control group statement.

When this parameter is specified, it replaces the specified map in the existing directory. The map will be added to the directory if it does not exist.

The CTRAN, CDIR, and SLIMIT parameters, if specified, will replace the corresponding values in the existing directory. For the description of the function of these parameters, see Defining Directories on page 4-45.

Notes:

- The NAME parameter plus at least one of the other parameters must be used.
- Only one map can be updated or added per each UPDATEDIR command. To update or add more than one map, use multiple UPDATEDIR commands.
- The EXCLUDE statement as described in Defining Directories on page 4-45, can be used with the UPDATEDIR statement to remove commands from the vocabulary specified in the VOCAB parameter, or to remove fields or segments from the map specified in the MAP parameter.

- Updating a map in a directory requires all the stored inquiries and functions for the updated map in that directory and any directory connected to it to be restored again. You can use the IXUSQRY utility to unload and then restore the stored inquiries for the updated map.
- A map can only be added to a directory which is not connected to any other directory. Trying to add a map to a connected directory will terminate the process and an error message will be issued. Example:

  DIRECT1 ========> connected to ========>DIRECT2

  A map can be added to DIRECT2, but not to DIRECT1.
- The CTRAN and CDIR parameters may be left blank (CTRAN=, CDIR=) which disconnects the connection for the directory.
- When updating CTRAN and CDIR parameters of the existing directory to connect to a different directory or to remove the connection, you should consider the stored inquiries and functions of the existing directory carefully. Trying to run the stored inquiries and functions which reference databases or files from the old connected directory that does not exist in the new connection may cause unpredictable results.
- The coding rules described in IIGEN Coding Rules on page 4-7, are also applicable to the UPDATEDIR statement.

UPDATEDIR Control Statement Group

The UPDATEDIR control statement group is terminated by a standard FINISH statement that contains no parameters.

The UPDATEDIR control statement group must be included within an APPL/END control statement group to effect the update of directories for that application (APPL) only.

Example 1:

```
APPL  NAME=IQIO
UPDATEDIR  NAME=IIDMDIR,VOCAB=VOCAB1,SLIMIT=1000
FINISH
END
```

Replaces the existing vocabulary of the directory IIDMDIR with vocabulary VOCAB1 for the application IQIO. It also updates the SLIMIT value with 1000 for the directory.

Example 2:

```
APPL  NAME=IQIO
UPDATEDIR  NAME=IIDMDIR,VOCAB=VOCAB2
EXCLUDE  VOCAB=VOCAB2,NAME=OUTPUT,NAME=EXTRACT
FINISH
END
```

Replaces the existing vocabulary of the directory IIDMDIR with vocabulary VOCAB2 for the application IQIO. It also excludes the commands OUTPUT and EXTRACT from the replaced vocabulary.

Example 3:

```
APPL  NAME=IQIO
UPDATEDIR  NAME=IIDMDIR,MAP=IIDMMAP
EXCLUDE  MAPNAME=IIDMMAP,NAME=SAL.DED
FINISH
```

Replaces or adds (if it does not exist) MAP IIDMMAP in directory IIDMDIR for application IQIO. It also excludes the field SAL.DED from the replaced or added map.

DELETE Statement

Use the DELETE statement of IIGEN to remove the following from the system database:

- Maps
- Directories
- TERMs
- Vocabularies
- All the elements of an application

The format of the DELETE statement is:

```
DELETE   DIRECTORY=dir-name)(,TERM=term-name)
         (,MAP=map-name)(,VOCAB=vocab-name)(,ALL)
```

The DELETE statement parameters are:

DIRECTORY= dir-name	Represents a valid directory name to be deleted from the application. Additionally, all TERMs or generic TERMs associated with this directory are also deleted. Any connected directory to the deleted directory must be re-entered in the system database.
TERM= term-name	Specifies the name of a TERM to be deleted.
MAP= map-name	Specifies the name of a database map to be deleted. Any directory containing the deleted map and any connected directory to it must be re-entered in the system database.
VOCAB= vocab-name	Specifies the name of a vocabulary to be deleted. The DELETE statement must be specified within the APPL $$$$$IXX.
ALL	The entire application is deleted (causing all the directories, TERMs, and maps within the application to be deleted). Any connected directory to directories of this application must be re-entered in the system database.

Note: You can associate as many parameters as needed with the DELETE statement. The only restrictions are the rules discussed in IIGEN Coding Rules on page 4-7.

DELETE Control Statement Group

The DELETE control statement group is terminated by a standard FINISH statement that contains no parameters.

The DELETE control statement group must be included within an APPL/END control statement group to effect the deletion of elements for that application (APPL) only. Failure to do so could damage the system database.

Example:

```
APPL NAME=IQIO
DELETE DIRECTORY=PAY1,MAP=MAPA
FINISH
END
```

5 The Utilities

This chapter describes the VISION:Inquiry utility programs for creating and maintaining the system database, and the VISION:Journey download data set.

Before using any of the utilities in this chapter, you should read or be familiar with the information describing the organization and contents of the databases, found in Chapter 3, "System Components".

Introducing the Utilities

The utilities described in this chapter are:

IIINIT	Initializes the system database.
IIGEN	Creates and maintains VISION:Inquiry error and diagnostic messages, vocabularies, maps, directories, and TERMs.
IXUUNLD	Unloads the system database.
IXULOAD	Loads the system database.
IXUSTAT	Obtains system database statistics.
IXUSQRY	Unloads the source of stored inquiries and functions.
IXUIQRY	Converts the stored inquiries and functions from internal to source format and saves them in a sequential data set. This utility is not applicable to native SQL syntax stored inquiries which are stored in source format only in the system database.
IFUCLEN	Deletes unwanted extracted data from the VISION:Journey download data set.
VSMFTSRE	JCL to reorganize the VISION:Journey VSAM download data set to eliminate the CI/CA splits.

Utility Input Command Statements

Selected utilities require input command statements in an 80-byte record format. The input command statements you specify contain parameters that direct the utilities to perform specified functions.

Utility	Input Command Statements
IIINIT	INDEX/DIRECTORY statement
IIGEN	In addition to the IIGEN statements described in Chapter 4, "The Definition Process", the following statements are also used with the IIGEN utility: ERRLOAD, ERRMSG, and ERRHELP statements MSGLIST statement SYSTEM and SYSLOAD statements
IXUUNLD	None
IXULOAD	OPTIONS statement with INCLUDE and EXCLUDE statements
IXUSTAT	None
IXUSQRY	APPL, MAP, or ALL statements The statements for the IXUSQRY utility do not have commands; the parameters may begin in any column.
IXUIQRY	None
IFUCLEN	OPTIONS statement

Coding Conventions

When preparing input command statements, use the following conventions:

- Each statement must contain a command, with no embedded blanks, that identifies the statement type.
- Each statement can contain keyword parameters. These parameters can begin in any column to the right of the blank following the command. They can be in any order but must be separated by commas with no embedded blanks.
- Each parameter must be coded once, unless stated otherwise.
- A comma following the last parameter on a statement causes a continuation to the next statement.
- Comments can be placed after the last parameter with an intervening blank.
- Columns 1 through 71 of the input are scanned; columns 72 through 80 are ignored.
- An asterisk (*) in column 1 followed by comments can be entered at any point in the input.
- A 1 in column 1 causes the page to eject when the output is listed.

Notation

The following notations are used in describing the command statements:

- Brackets [ ] are used to indicate that the enclosed items are optional and may be omitted.
- Braces { } are used to group related items that are alternatives. One item must be chosen. The context indicates when choosing more than one item is appropriate.
- A default value within a group of items is underlined.
- The commands, keyword parameters, and default values are shown in upper-case letters.
- Parameter values you provide are shown in lower-case letters.

IIINIT Utility

The IIINIT utility allocates and initializes the system database. It formats database segments and constructs dummy index segments/records.

The IIINIT utility uses the INDEX/DIRECTORY statement as input. The two parameters specify the number of database segments/records and index segments/records to be formatted. The segments/records initialized by IIINIT are the only ones used by VISION:Inquiry; no new segments/records are added and no old segments/records deleted.

IIGEN Utility

The IIGEN utility provides the means for defining applications and generating system-wide requirements. The IIGEN utility functions (discussed in this chapter) are:

- Create and modify error and diagnostic messages.
- Create and modify system and user vocabularies.

The IIGEN utility input depends on the function being performed:

- ERRLOAD control statement groups with ERRLOAD, ERRMSG, and ERRHELP statements are input for system database messages
- SYSTEM control statement groups with SYSTEM and SYSLOAD statements are input for system database vocabularies.

Other IIGEN functions, such as the definition of applications (MAPGENs, directories, and TERMs) are described in Chapter 4, "The Definition Process".

IXUUNLD Utility

The IXUUNLD utility creates an unloaded copy of a system database by copying each database element to a sequential, variable length record data set.

The IXUUNLD utility, in conjunction with IXULOAD, should be used in preference to general-purpose system copy or backup utilities, because the VISION:Inquiry utilities perform maintenance and reorganization of the internal structure of the system database.

There are no input statements for this utility.

IXULOAD Utility

The IXULOAD utility reloads a system database by copying the database elements from the unloaded sequential data set written by IXUUNLD.

The system database that receives the unloaded elements must have been initialized using the IIINIT utility prior to reloading. IXULOAD normally restores all the database elements; however, through optional input commands, you may control which elements are reloaded.

The IXULOAD utility uses the OPTIONS control statement group with the OPTIONS, INCLUDE, and EXCLUDE statements as input.

IXUSTAT Utility

The IXUSTAT utility provides statistical information about the components and their space utilization on the system database. You can use this information to determine the optimum organization of the applications, maps, and directories to minimize the number of system database calls or reads made by the inquiry processing programs. By running IXUSTAT periodically, you can monitor the amount of space being used by stored functions, stored inquiries, and deferred inquiries.

There are no input statements for this utility.

IXUSQRY Utility

The IXUSQRY utility creates an unloaded copy of the source of stored inquiries and functions by copying the stored inquiries and functions to a sequential, 80-byte fixed length record data set. The sequential data set can then be used as input to the batch version of the product to restore the inquiries and functions back to the system database. The batch version of the product uses the sample JCL in the II.TCVLCNTL library members IQBATD (for an IMS system database) or IQBATV (for a DB2 or VSAM system database).

The stored inquiries and functions can be unloaded through the input statements for all applications, a specific map, or a specific directory or map within an application.

The IXUSQRY utility uses an APPL, MAP, or ALL statement as input.

IXUIQRY Utility

The IXUIQRY utility is used to upgrade inquiries and functions from releases prior to Release 6.0. It converts the stored inquiries and functions from internal to source format and saves them in a sequential, 80-byte fixed length record data set. This data set can then be used as input to the batch version of the product to restore the inquiries and functions to the system database. The batch version of the product uses the sample JCL in the II.TCVLCNTL library members IQBATD (for an IMS system database) or IQBATV (for a DB2 or VSAM system database).

This utility is not applicable to native SQL syntax stored inquiries which are stored in source format only in the system database.

There are no input statements for this utility.

IFUCLEN Utility

The IFUCLEN utility is used to delete the unwanted data extracted and written to the VISION:Journey download data set.

The IFUCLEN utility uses the OPTIONS control statement group as input to specify which data should be deleted.

Accessing the System Database

The VISION:Inquiry utilities IIINIT, IIGEN, IXUUNLD, IXULOAD, IXUSTAT, IXUSQRY, and IXUIQRY access the system database. The JCL requirements differ depending on whether you have an IMS (DL/I), VSAM, or a DB2 system database.

Model JCL for executing the system database utilities can be found in II.TCVLCNTL members.

Utility	II.TCVLCNTL members for an IMS (DL/I) system database	II.TCVLCNTL members for a DB2 system database	II.TCVLCNTL members for a VSAM system database
IIINIT	IMSINIT	DB2INIT	VSAMINIT
IIGEN	IMSELEM	DB2ELEM	VSAMELEM
IXUUNLD IXULOAD	IMSUNLD IMSLOAD	DB2UNLD DB2LOAD	VSAMUNLD VSAMLOAD
IXUIQRY IXUSQRY	IMSIQRY IMSSQRY	DB2IQRY DB2SQRY	VSAMIQRY VSAMSQRY
IXUSTAT	IMSSTAT	DB2STAT	VSAMSTAT

This chapter contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing "DB2" is specifically applicable to sites licensed with the DB2 option.

Access an IMS (DL/I) System Database

To access an IMS (DL/I) system database, utility programs are run using the DLIBATCH procedure. (IIGEN can also be run as a BMP, using the IMSBATCH procedure.)

- The name of a PSB containing a PCB for the system database is specified as a parameter to the procedure.
- II.TCVLSRC (IIPSB01) contains an example of the load PSB required by IIINIT.
- II.TCVLSRC (IIPSB02) is appropriate for the other utilities.

Whenever DLIBATCH is used, the first database PCB in the PSB represents the system database. Ensure that the corresponding DD statement references the system database you wish to use.

Access a DB2 System Database

To access a DB2 system database, utility programs are run in an ordinary batch region.

The load modules needed for CALL Attach must be available in the STEPLIB concatenation or the link list (including DSN510.SDSNLOAD) for successful access to DB2 data.

Access a VSAM System Database

For a VSAM system database, utility programs are run in an ordinary batch region.

A DD statement referencing the VSAM cluster used as system database must be supplied. The ddname is the value, if any, specified for the NAME operand in IXSIDENT; otherwise, the ddname is IXXDB.

Access a VISION:Journey VSAM Download Data Set

For the VISION:Journey VSAM download data set, the following utilities are used:

- The model JCL for executing the utility, IFUCLEN, is VSMFTSCL in II.TCVLCNTL. The utility program is run in an ordinary batch region. During installation of the product, specify the appropriate VSAM file ddname and password (if necessary) so that the utility can access the VISION:Journey VSAM download data set.
- The II.TCVLCNTL member VSMFTSIN initializes the VISION:Journey VSAM download data set.
- The II.TCVLCNTL member VSMFTSRE reorganizes the VISION:Journey VSAM download data set.

Initializing the System Database with IIINIT

The database you specify to IIINIT as your VISION:Inquiry system database is initialized by formatting segments and constructing dummy index segments/records.

INDEX/DIRECTORY Statement

The number of segments/records to be initialized are specified through the INDEX/DIRECTORY statement. This is the only statement required for IIINIT to execute. This statement derives its name from the two parameters that comprise it, INDEX and DIRECTORY.

The format of the INDEX/DIRECTORY statement is:

INDEX=ni,DIRECTORY=nd;

The INDEX/DIRECTORY statement parameters are:

INDEX=ni	Specifies the number of segments/records to be formatted for the database index. The maximum is 250.
DIRECTORY=nd	Specifies the number of segments/records to be formatted in addition to the index. The maximum is 32320.
;	The semicolon is the required delimiter.

IIINIT JCL Requirements

Sample JCL to allocate and initialize an IMS (DL/I) system database is found in control library member II.TCVLCNTL (IMSINIT). A listing of this member is given in the Defining and Initializing the System Database section of the *Advantage* VISION:Inquiry *for CICS Installation Guide*. The ddname used for the system database must match that specified in the DBD.

Sample JCL to initialize a DB2 system database is found in the control library member II.TCVLCNTL (DB2INIT). A listing of this member is given in the Defining and Initializing the System Database section of the *Advantage VISION:Inquiry for CICS Installation Guide*.

Sample JCL to allocate and initialize a VSAM system database is found in control library member II.TCVLCNTL(VSAMINIT). A listing of this member is given in the Defining and Initializing the System Database section of the *Advantage VISION:Inquiry for CICS Installation Guide.*

Defining Messages for AQF with IDBERRS

When AQF is being used, most error messages are in the mapset IDBERRSM. The source for this mapset can be found in the member IDBERRS in II.TCVLSRC. You are free to alter the message contents (but at most one line per message).

You can also have different versions of IDBERRS for different users by either assigning their terminal different alternate suffixes.

Diagnostic messages issued by AQF are in the following format:

AQF-nnn text

Where:

AQF-nnn	Specifies the unique 3-digit identifier for this error or diagnostic message.
text	Describes the error or diagnostic condition.

The AQF messages appear on the last line of the screen.

Defining VISION:Inquiry System Database Elements with IIGEN

After the system database is initialized by IIINIT, use IIGEN to define the vocabularies, error messages, applications, MAPGENs, directories, and logical terminal descriptions.

This section describes the functions of IIGEN that are related to the elements used on a system-wide basis. These functions are:

- System database diagnostic messages - Defining Native VISION:Inquiry Messages on page 5-11
- System database vocabularies - Defining Vocabularies with IIGEN on page 5-16

Other IIGEN functions, such as the definition of applications (MAPGENs, directories, and TERMs) are described in Chapter 4, "The Definition Process".

Defining Native VISION:Inquiry Messages

Most native VISION:Inquiry messages that are issued by IIGEN and the inquiry processing programs are kept in the system database. Messages are independent of the programs and easily modified to suit your needs, for example, to translate them into different languages.

- A standard set of error and diagnostic messages, comprising all messages stored in the system database, is included in II.TCVLSRC (IIERROR).
- Native VISION:Inquiry messages and hard-coded messages are contained in the *Advantage VISION:Inquiry Messages Guide*.
- If desired, the text of the messages stored in the system database can be altered. This includes lengthening or shortening the message text and translating the message into a different language.

There can only be one set of diagnostic messages per system database.

The format of a native VISION:Inquiry message is:

xxxyyyy text

where:

xxx	Distinguishes error messages issued by the batch utility programs (IXG) from those of the inquiry processing programs (IXX).
yyyy	Specifies the unique 4-digit numeric identifier for this error or diagnostic message.
text	Describes the error or diagnostic condition.

Help text has additional information, listing some of the possible reasons for the error or providing a description of valid specifications regarding the erroneous item.

ERRLOAD Control Statement Group

IIGEN uses ERRLOAD control statement groups to generate error and diagnostic messages.

An ERRLOAD control statement group consists of the following:

```
ERRLOAD Statement
        ERRMSG Statement
                ERRHELP Statement (optional)
                        .
                        .
                        .
        ERRMSG Statement
                ERRHELP Statements (optional)
                        .
                        .
                        .
FINISH Statement
END Statement
```

Each ERRMSG statement describes one error or diagnostic message.

Each ERRHELP statement describes one line of Help text.

The ERRLOAD control statement group must not be included within an APPL/END control statement set. The entire control statement group must end with a standard FINISH statement and an END statement.

ERRLOAD Statement

The ERRLOAD statement defines the beginning of the group and must be the first statement of the group. The ERRLOAD statement must precede any APPL/END groups.

The format of the ERRLOAD statement is:

ERRLOAD

No parameters are required. Comments can be coded after ERRLOAD and at least one intervening blank.

ERRMSG Statement

The ERRMSG statement defines one error or diagnostic message.

The format of the ERRMSG statement is:

ERRMSG “aaaabcc text INCLUDING SUBSTITUTION’#’VARIABLES”

The ERRMSG statement parameters are:

“ Specifies the left hand delimiter of the error information (double quotation mark).

aaaa Specifies the unique 4-digit numeric identifier for the error or diagnostic message.

b Specifies a code (0 or 1) that indicates whether or not sequence numbers are to be displayed for input records detected to have errors by the batch utility programs. This includes only messages 0001 through 0099.

- A 0 means that sequence numbers should not be displayed.
- A 1 indicates that they should be displayed.

cc Specifies the severity level of the error or diagnostic messages being issued by the batch utility programs. Again, this includes only messages 0001 through 0099.

text Specifies the text to be used for this error condition. Substitutable variables are contained within the text of some error messages as indicated by ’#’.

At this point in the message, VISION:Inquiry inserts the appropriate data to qualify the error detected. It may be a field name, an arithmetic operator, a function name, or some other element name of the system that clarifies the message being issued.

Users can, at their discretion, eliminate this substitutable data within certain messages for security or other reasons.

- The text length with the insert cannot exceed terminal width.
- Most of the standard error and diagnostic messages are less than 70 bytes.
- They can be expanded to a maximum length of 240 bytes, if desired.

Follow the syntax rules for utility input.

” Specifies the right hand delimiter of the error information (double quotation mark).

ERRHELP Statement

The ERRHELP statement defines one line of Help text.

The format of the ERRHELP statement is:

ERRHELP “aaaa help text”

The ERRHELP statement parameters are:

“	Specifies the left hand delimiter of the help information (double quotation mark).
aaaa	Specifies a 4-digit numeric identifier that must match the preceding ERRMSG statement value.
help text	Specifies one line of the informational text to be displayed following the related error message. The maximum length of one line of text is 240 characters; text exceeding the terminal width value is truncated on output. For each message, a maximum of 20 lines may be specified, and a maximum of 960 characters of text may be specified (Message text and Help text combined).
”	Specifies the right hand delimiter of the help information (double quotation mark).

All of these parameters must be specified for initially defining or subsequently altering error and diagnostic messages.

Only one error and diagnostic message can be altered per ERRMSG statement. As many ERRMSG and ERRHELP statements as described can be entered in the ERRLOAD control statement group.

The system error and diagnostic messages are generated from the ERRLOAD control statement group found in II.TCVLSRC (IIERROR). This should be referenced as a guideline when altering messages.

ERRLOAD builds all the error and diagnostic messages and you must account for every message. If you were to code only one ERRMSG statement, only that message would have a text. All others would be “No Text”.

Figure 5-1 illustrates a partial sample error ERRLOAD control statement group.

```
*                   STANDARD ERROR MESSAGES
ERRLOAD
ERRMSG   "0001108 THE CONTROL COMMAND '#' CANNOT BE IDENTIFIED."
ERRMSG   "0002108 DATABASE MAP '#' HAS NO SEGMENTS."
ERRMSG   "0003108 DATABASE/VSAM MAP '#' HAS NO FIELDS."
ERRMSG   "0004108 ILLEGAL OR MISSING PARAMETER FOR '#' SPECIFICATION."
ERRMSG   "0005108 '#' IS AN ILLEGAL SYSTEM NAME."
ERRMSG   "0006104 NO NAME SPECIFIED FOR MAPGEN, DBD NAME '#' USED."
ERRMSG   "0007104 FINISH COMMAND MISSING FOR DATABASE/VSAM MAP '#'."
ERRMSG   "0008104 NO VOCABULARY NAME SPECIFIED 'SYSVOCAB' IS ASSUMED."
ERRMSG   "0009108 REFERENCED SEGMENT '#' NOT FOUND. "
ERRMSG   "0010108 UNEXPECTED END OF FILE ON SYSIN. EXPECTED '#'."
ERRHELP  "0010 A 'FINISH' AND/OR 'END' STATEMENT WAS NOT ENCOUNTERED PRIOR TO END OF FILE."
ERRHELP  "0010 'FINISH' MUST TERMINATE EACH CONTROL STATEMENT GROUP(MAPGEN,LTERM,ERRLOAD,ETC.)."
ERRHELP  "0010 'END' MUST BE THE LAST STATEMENT IN A 'APPL/END' OR 'ERRLOAD/END' SET."
ERRHELP  "0010 THE MISSING STATEMENT WAS SUPPLIED BY VISION:Inquiry"
ERRHELP  "0010 HOWEVER, YOU SHOULD CHECK THE RESULT CLOSELY TO INSURE THAT IT IS CORRECT."
ERRMSG   "0011000 # ERROR MESSAGES WERE LOADED INTO THE SYSTEM DATABASE."
ERRMSG   "0012000 # SYSTEM DATABASE RECORDS WERE REQUIRED TO STORE THE ERROR MESSAGES."
ERRMSG   "0013108 CONTINUATION OF STATEMENT WAS EXPECTED WHEN END OF FILE OCCURRED. STATEMENT
               PROCESSED AS IS."
ERRMSG   "0014108 EXPECTED A COMMAND BUT FOUND #. SYSIN RECORD DISCARDED."
ERRMSG   "0015108 BODY OF STATEMENT EXCEEDS MAX SIZE OF 2048 CHARACTERS BY #. STATEMENT DISCARDED."
ERRMSG   "0016108 FINISH COMMAND MISSING FOR DIRECTORY '#'."
ERRMSG   "0017108 COMMAND '#' NOT VALID IN DIRECTORY GROUP. GROUP TERMINATED."
ERRMSG   "0018108 CONTROL CARD ERROR, DATABASE/VSAM MAP '#' NOT CREATED."
ERRMSG   "0019108 '#' COMMAND STATEMENT DISCARDED BECAUSE AN 'APPL' GROUP IS BEING PROCESSED."
ERRMSG   "0020108 '#' COMMAND STATEMENT DISCARDED BECAUSE NO VALID 'APPL' GROUP IS BEING PROCESSED."
ERRMSG   "0021108 REQUIRED LTERM/TERM NAME MISSING."
ERRMSG   "0022108 SORT LIMIT IS GREATER THAN 32767."
ERRMSG   "0023108 FIELD CANNOT BE CONTAINED WITHIN ESTABLISHED KEY LENGTH."
ERRMSG   "0024108 AN ENGLISH NAME WAS DEFINED MORE THAN EIGHT TIMES FOR THIS ENTRY, LIMIT EXCEEDED."
ERRMSG   "0025108 'TERMWDTH' MUST BE 1 TO 4 DIGITS. FOUND '#'."
ERRMSG   "0026108 'TERMLTH' MUST BE 1 TO 4 DIGITS. FOUND '#'."
ERRMSG   "0027108 FIELD SPECIFIED EXCEEDS THE TOTAL LENGTH SPECIFIED FOR THE SEGMENT."
ERRMSG   "0028108 THE SEGMENT NAME '#' CANNOT BE IDENTIFIED."
```

Figure 5-1 A Sample ERRLOAD Control Statement Group

MSGLIST Statement

The MSGLIST statement lists the messages in the system database and can be used at any time after the error and diagnostic messages are built. The format of the MSGLIST statement is:

The format of the MSGLIST statement is:

```
MSGLIST
END
```

Defining Vocabularies with IIGEN

The VISION:Inquiry language consists of the following:

- Commands
- Relational Operators
- Arithmetic Operators
- Symbols
- Noise words

These elements are not defined in the VISION:Inquiry programs, but kept separately in the system database in one or more vocabularies. By keeping vocabularies in a system database, they are readily customized and easily adapted to different languages.

The manner in which a system vocabulary is described and created lends itself to a high degree of flexibility in the selection of words to be used in the vocabulary. Each word in the vocabulary is represented as a code that is known to VISION:Inquiry. These words are translated into codes that are actions for VISION:Inquiry.

VISION:Inquiry provides the capability for creating as many different vocabularies as desired.

Each system vocabulary can contain whatever words and existing functions that you want VISION:Inquiry to recognize. This type of independence is easily conceptualized by thinking of the command, operators, and so on, as functions. Each function is identified by arbitrary sets of words that form a system vocabulary.

The system must contain at least one vocabulary. A standard set of vocabulary entries is provided with the VISION:Inquiry system and installed as part of the installation process.

SYSTEM Control Statement Group

A vocabulary is generated by a SYSTEM control statement group.

The SYSTEM control statement group consists of the following statements:

```
SYSTEM Statement
        SYSLOAD Statement
        .
        .
        .
        SYSLOAD Statement
FINISH Statement
END Statement
```

Every vocabulary is defined with the application $$$$$IXX. The SYSTEM statement must be preceded by:

```
APPL NAME=$$$$$IXX
```

SYSTEM Statement

The SYSTEM statement defines the beginning of a vocabulary and must be the first statement of the group.

The format of the SYSTEM statement is:

SYSTEM VOCAB={SYSVOCAB|name}

The SYSTEM statement parameter is:

VOCAB=name Defines a 1 to 8 character unique name that is used to identify this vocabulary. If the VOCAB parameter is omitted, the value SYSVOCAB is assumed by default.

SYSLOAD Statement

The SYSLOAD statement defines one word to be included in a vocabulary.

When creating a user vocabulary, you specify a SYSLOAD statement for each word (from the system vocabulary) that is to be included in the user vocabulary.

The format of the SYSLOAD statement is:

SYSLOAD WORD=name,CODE=code

The SYSLOAD statement parameters are:

WORD=name Specifies the word to be defined.

CODE=code Specifies the 4-digit code that describes the function to VISION:Inquiry that this word will reference.

Codes and their meanings are listed in Appendix A, "System Vocabulary and Codes". Synonyms can be created by defining each one with the same code. It is important to remember that every code in the standard system vocabulary must be included when you create your vocabulary. Any function not included is unavailable to users of the vocabulary.

Both parameters are required and both can be specified only once per statement.

VISION:Inquiry allows the use of 'noise words' in its language. Noise words are words that are ignored by the system, but which might make an inquiry more readable to a user. All noise words must have a code of 0100.

Inclusion of extra codes results in errors when trying to use VISION:Inquiry.

Required Commands

The Text Editor and the utilities, IXUSQRY, IXUIQRY, and the Intraccess option, require some commands to be present in the vocabulary which is created and included in the directories. These commands are:

- DDI and EDITSQ commands for the Text Editor
- DDF and DDI commands for IXUSQRY
- The standard vocabulary supplied in II.TCVLSRC (IIVOCAB) for IXUIQRY and the Intraccess option

However, you may define synonyms for the words of the supplied vocabulary if you want users to use different words.

Sample Vocabulary

Figure 5-2 illustrates a SYSTEM control statement group for a sample vocabulary.

```
APPL            NAME=$$$$$IXX
SYSTEM          VOCAB=SYSVOCAB
SYSLOAD         WORD=AS,CODE=0100
SYSLOAD         WORD=A,CODE=0100
SYSLOAD         WORD=AN,CODE=0100
SYSLOAD         WORD=ARE,CODE=0100
SYSLOAD         WORD=BY,CODE=0100
SYSLOAD         WORD=FROM,CODE=0100
SYSLOAD         WORD=HAD,CODE=0100
SYSLOAD         WORD=HAS,CODE=0100
SYSLOAD         WORD=HAVE,CODE=0100
SYSLOAD         WORD=IN,CODE=0100
SYSLOAD         WORD=IS,CODE=0100
SYSLOAD         WORD=NO,CODE=0100
SYSLOAD         WORD=NONE,CODE=0100
SYSLOAD         WORD=OF,CODE=0100
SYSLOAD         WORD=ON,CODE=0100
SYSLOAD         WORD=THAN,CODE=0100
SYSLOAD         WORD=THE,CODE=0100
SYSLOAD         WORD=TO,CODE=0100
SYSLOAD         WORD=WAS,CODE=0100
SYSLOAD         WORD=WERE,CODE=0100
SYSLOAD         WORD=ASC,CODE=5600
SYSLOAD         WORD=DSC,CODE=5680
SYSLOAD         WORD=DISPLAY,CODE=1300
SYSLOAD         WORD=D,CODE=1300
SYSLOAD         WORD=FIND,CODE=1100
SYSLOAD         WORD=WITH,CODE=3000
SYSLOAD         WORD=IF,CODE=3000
SYSLOAD         WORD=ALL,CODE=3080
SYSLOAD         WORD==,CODE=5000
SYSLOAD         WORD=EQ,CODE=5000
SYSLOAD         WORD=EQUAL,CODE=5000
SYSLOAD         WORD=<,CODE=5004
SYSLOAD         WORD=LT,CODE=5004
SYSLOAD         WORD=>,CODE=5008
SYSLOAD         WORD=GT,CODE=5008
SYSLOAD         WORD=<=,CODE=5010
SYSLOAD         WORD=LE,CODE=5010
SYSLOAD         WORD=>=,CODE=5014
```

Figure 5-2 SYSTEM Control Statement Group for a Sample Vocabulary (Page 1 of 2)

```
SYSLOAD          WORD=GE,CODE=5014
SYSLOAD          WORD=¬=,CODE=5018
SYSLOAD          WORD=NE,CODE=5018
SYSLOAD          WORD=FIRST,CODE=5100
SYSLOAD          WORD=LAST,CODE=5200
SYSLOAD          WORD=AND,CODE=6000
SYSLOAD          WORD=&,CODE=6000
SYSLOAD          WORD=OR,CODE=6100
SYSLOAD          WORD=;,CODE=6000
FINISH           SYSLOAD
END
```

Figure 5-2 SYSTEM Control Statement Group for a Sample Vocabulary (Page 2 of 2)

IIGEN JCL Requirements

Sample JCL to execute IIGEN using an IMS (DL/I) system database is found in control library member II.TCVLCNTL (IMSELEM). For a listing, see the Defining a Test Application section of the *Advantage VISION:Inquiry for CICS Installation Guide*.

Sample JCL to execute IIGEN using a DB2 system database is found in control library member II.TCVLCNTL (DB2ELEM). For a listing see the Defining a Test Application section of the *Advantage VISION:Inquiry for CICS Installation Guide*.

Sample JCL to execute IIGEN using a VSAM system database is found in control library member II.TCVLCNTL(VSAMELEM). For a listing see the Defining a Test Application section of the *Advantage VISION:Inquiry for CICS Installation Guide.*

IIGEN Output

The output listing produced by IIGEN consists of two parts: a listing of the input stream, and a listing of diagnostic messages. As the input stream is processed, input statements are numbered. Figure 5-3 on page 5-21 illustrates a typical IIGEN output listing.

All diagnostic messages appear separately from the input statements. There are both informational and error messages. These messages inform you of any errors encountered during processing and of the action taken. Error messages for IIGEN are described in the *Advantage VISION:Inquiry Messages Guide*.

Each message is printed with an accompanying severity level that reflects error conditions or actions taken. The highest severity level encountered is returned to the operating system as a completion code. If the highest severity level is 12 or greater, the system database may be full or damaged.

The following severity levels appear with a diagnostic message:

0 Information message that documents a successful action.

4 Warning message indicating a potential error.

8 Error message indicating an error condition occurred during command statement processing. The action specified for the command is not taken.

12 Error message indicating that a serious error has occurred. Further processing is not taken.

16 Terminating error.

```
12/26/2002 VISION:Inquiry 6.5 FOR CICS SYSTEM DEFINITION AND GENERATION DIAGNOSTICS

  *       STANDARD ERROR MESSAGES
  1       ERRLOAD
  2       ERRMSG    "0001108 THE CONTROL COMMAND '#' CANNOT BE IDENTIFIED."
  3       ERRMSG    "0002108 DATABASE MAP '#' HAS NO SEGMENTS."
  4       ERRMSG    "0003108 DATABASE/VSAM MAP '#' HAS NO FIELDS."
  5       ERRMSG    "0004108 ILLEGAL OR MISSING PARAMETER FOR '#' SPECIFICATION."
  6       ERRMSG    "0005108 '#' IS AN ILLEGAL SYSTEM NAME."
  7       ERRMSG    "0006104 NO NAME SPECIFIED FOR MAPGEN, DBD NAME '#' USED."
  8       ERRMSG    "0007104 FINISH COMMAND MISSING FOR DATABASE/VSAM MAP '#'."
  9       ERRMSG    "0008104 NO VOCABULARY NAME SPECIFIED 'SYSVOCAB' IS ASSUMED."
 10       ERRMSG    "0009108 REFERENCED SEGMENT '#' NOT FOUND. "
 11       ERRMSG    "0010108 UNEXPECTED END OF FILE ON SYSIN. EXPECTED '#'."
 12       ERRHELP   "0010 A 'FINISH' AND/OR 'END' STATEMENT WAS NOT ENCOUNTERED PX
                     RIOR TO END OF FILE."
 13       ERRHELP   "0010 'FINISH' MUST TERMINATE EACH CONTROL STATEMENT GROUP(MAX
                     PGEN,LTERM,ERRLOAD,ETC)."
 14       ERRHELP   "0010 'END' MUST BE THE LAST STATEMENT IN A 'APPL/END' OR 'ERX
                     RLOAD/END' SET."
 15       ERRHELP   "0010 THE MISSING STATEMENT WAS SUPPLIED BY VISION:Inquiry."
 16       ERRHELP   "0010 HOWEVER, YOU SHOULD CHECK THE RESULT CLOSELY TO INSURE X
                     THAT IT IS CORRECT."
 17       ERRMSG    "0011000 # ERROR MESSAGES WERE LOADED INTO THE SYSTEM DATABAX
                    SE."
 18       ERRMSG    "0012000 # SYSTEM DATABASE RECORDS WERE REQUIRED TO STORE THEX
                     ERROR MESSAGES."
 19       ERRMSG    "0013108 CONTINUATION OF STATEMENT WAS EXPECTED WHEN END OF FX
                     ILE OCCURRED. STATEMENT PROCESSED AS IS."
 20       ERRMSG    "0014108 EXPECTED A COMMAND BUT FOUND #. SYSIN RECORD DISCARDX
                     ED."
 21       ERRMSG    "0015108 BODY OF STATEMENT EXCEEDS MAX SIZE OF 2048 CHARACTERX
                     S BY #. STATEMENT DISCARDED."
 22       ERRMSG    "0016108 FINISH COMMAND MISSING FOR DIRECTORY '#'."
 23       ERRMSG    "0017108 COMMAND '#' NOT VALID IN DIRECTORY GROUP. GROUP TERMX
                     INATED."
 24       ERRMSG    "0018108 CONTROL CARD ERROR, DATABASE/VSAM MAP '#' NOT CREATX
                    ED."
 25       ERRMSG    "0019108 '#' COMMAND STATEMENT DISCARDED BECAUSE AN 'APPL' GRX
                     OUP IS BEING PROCESSED."
 26       ERRMSG    "0020108 '#' COMMAND STATEMENT DISCARDED BECAUSE NO VALID 'APX
                     PL' GROUP IS BEING PROCESSED."
 27       ERRMSG    "0021108 REQUIRED LTERM/TERM NAME MISSING."
 28       ERRMSG    "0022108 SORT LIMIT IS GREATER THAN 32767."


426          SYSLOAD   WORD=DDI,CODE=1813
427          SYSLOAD   WORD=PD,CODE=1316
428          SYSLOAD   WORD=PDM,CODE=1312
429          SYSLOAD   WORD=PDV,CODE=1315
```

Figure 5-3 Typical IIGEN Output (Page 1 of 2)

```
 430          SYSLOAD   WORD=PDI,CODE=1313
 431          SYSLOAD   WORD=PDF,CODE=1317
 432          SYSLOAD   WORD=PDL,CODE=1314
 433          SYSLOAD   WORD=DI,CODE=1A03
 434          SYSLOAD   WORD=CDI,CODE=1B33
 435          SYSLOAD   WORD=PS,CODE=1320
 436          SYSLOAD   WORD=FORMAT,CODE=0008
 437          SYSLOAD   WORD=PAGE,CODE=4000
 438          SYSLOAD   WORD=DATE,CODE=4001
 439          SYSLOAD   WORD=TIME,CODE=4002
 440          SYSLOAD   WORD=LINE,CODE=4003
 441          SYSLOAD   WORD=SKIP,CODE=4004
 442          SYSLOAD   WORD=SP,CODE=4005
 443          SYSLOAD   WORD=SPACE,CODE=4005
 444          SYSLOAD   WORD=COL,CODE=4006
 445          SYSLOAD   WORD=COLUMN,CODE=4006
 446          SYSLOAD   WORD=REP,CODE=4007
 447          SYSLOAD   WORD=REPEAT,CODE=4007
 448          SYSLOAD   WORD=EDIT,CODE=4008
 449          SYSLOAD   WORD=PF,CODE=4009
 450          SYSLOAD   WORD=NOSP,CODE=400A
 451          SYSLOAD   WORD=NOSPACE,CODE=400
 452          SYSLOAD   WORD=TEST,CODE=1D00
 453          SYSLOAD   WORD=SHOW,CODE=1E00
 454          SYSLOAD   WORD=FT,CODE=1D08
 455          SYSLOAD   WORD=FD,CODE=1308
 456          SYSLOAD   WORD=FI,CODE=1108
 457          FINISH    SYSLOAD
 458          END

12/26/2002 VISION:Inquiry 6.5 FOR CICS SYSTEM DEFINITION AND GENERATION DIAGNOSTICS

MESSAGE AT   LVL           MESSAGE
NUMBER  STMT

IXG0011      00 278 ERROR MESSAGES WERE LOADED INTO THE SYSTEM DATABASE.
IXG0012      00  12 SYSTEM DATABASE RECORDS WERE REQUIRED TO STORE THE ERROR MESSAGES.
IXG0049 457  00 THE SYSTEM VOCABULARY 'SYSVOCAB' HAS BEEN CREATED.

IXG9228      00 3 DIAGNOSTIC MESSAGES PRINTED.
IXG9229      00 HIGHEST SEVERITY LEVEL ENCOUNTERED = 0.
```

Figure 5-3 Typical IIGEN Output (Page 2 of 2)

Maintaining the System Database

The system database is extremely active. It is updated in batch by the IIGEN utility and whenever an inquiry is being executed in all environments. For this reason alone, it is recommended that you backup the database on a regular basis.

Whenever a checkpoint is taken during execution of VISION:Inquiry, information necessary to continue the inquiry is temporarily stored separately in a scratch pad area reserved for every TERM defined to VISION:Inquiry. If the inquiry is deferred, the information becomes permanent. This scratch pad area can quickly fill up if there are many deferred inquiries without an intervening 'CONTINUE DEFERRED INQUIRY' command. This prevents further use of VISION:Inquiry.

This condition can be alleviated by running the VISION:Inquiry utilities IXUUNLD and IXULOAD to free up all the terminal scratch pad areas.

The system database, saved by IXUUNLD, is restored by executing the IXULOAD utility. Inquiries and functions can be either included or excluded from the restored system database. This capability provides you with an administrative tool for managing stored definitions.

In order to successfully unload/reload the system database, the following steps must be executed:

1. Unload the system database using the utility IXUUNLD.
2. If more physical space is needed, scratch and re-allocate the system database.
3. Initialize the system database using the utility IIINIT.
4. Reload the system database using the utility IXULOAD.

Unloading the System Database with IXUUNLD

The IXUUNLD utility program unloads each element of the system database to a sequential, variable length record data set. There are no input control statements to the program. The SYSUT1 DD statement defines the output sequential data set.

IXUUNLD JCL

Use one of the following, as appropriate to the system database:

- II.TCVLCNTL (IMSUNLD)
- II.TCVLCNTL (DB2UNLD)
- II.TCVLCNTL (VSAMUNLD)

II.TCVLCNTL (IMSUNLD)

Control library member II.TCVLCNTL (IMSUNLD) contains the following sample JCL for unloading an IMS (DL/I) system database.

```
//* IMS SYSTEM DATABASE BACKUP JCL:
//*  -THIS JOB WILL PRODUCE A BACKUP OF AN IMS SYSTEM DATABASE.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE AND THE DBDS AND
//*     PSBS GENERATED.
//*
//*
//IXUUNLD EXEC DLIBATCH,MBR=IXUUNLD,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
//SYSUT1   DD DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092),
//            UNIT=TAPE,DISP=(NEW,CATLG,DELETE),
//            DSN=II.IXXDB.UNLOAD
```

Figure 5-4 Sample JCL to Unload an IMS (DL/I) System Database

II.TCVLCNTL (DB2UNLD)

Control library member II.TCVLCNTL (DB2UNLD) contains the following sample JCL for unloading a DB2 system database.

```
//* DB2 SYSTEM DATABASE BACKUP JCL:
//*  -THIS JOB WILL PRODUCE A BACKUP OF A DB2 SYSTEM DATABASE.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES
//*     IN THE PARM FIELD; OR YOU MAY OMIT THE PARM ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*
//*
//IXUUNLD EXEC PGM=IXUUNLD,REGION=2M,
//         PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//         DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092),
//            UNIT=TAPE,DISP=(NEW,CATLG,DELETE),
//            DSN=II.IXXDB.UNLOAD
```

Figure 5-5 Sample JCL to Unload a DB2 System Database

II.TCVLCNTL (VSAMUNLD)

Control library member II.TCVLCNTL (VSAMUNLD) contains the following sample JCL for unloading a VSAM system database:

```
//* VSAM SYSTEM DATA BASE BACKUP JCL:
//*   - THIS JOB WILL PRODUCE A BACKUP OF A VSAM SYSTEM DATA BASE.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DD NAME "IXXDB" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*
//*
//IXUUNLD  EXEC PGM=IXUUNLD,REGION=500K,
//         PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092),
//            UNIT=TAPE,DISP=(NEW,CATLG,DELETE),
//            DSN=II.IXXDB.UNLOAD
```

Figure 5-6 Sample JCL for Unloading a VSAM System Database

Reloading the System Database with IXULOAD

The IXULOAD utility program reloads the system database from the data set created by IXUUNLD. The SYSUT1 DD statement defines this data set.

- The reload program must be run only against a system database which has been newly initialized using IIINIT.
- The system database to be reloaded need not be of the same type (IMS, DB2, or VSAM) as the system database which was unloaded.
- In addition, by specifying IXULOAD input commands, you can optionally control which inquiries and functions are reloaded. If no input commands other than END are specified, the entire unloaded database is restored.

The input commands for the reload utility conform to the same coding conventions as those of IIGEN. (For more information, see Utility Input Command Statements on page 5-2 earlier in this chapter, and IIGEN Coding Rules on page 4-7.) Input command statements are specified in an OPTIONS control statement group.

OPTIONS Control Statement Group

The OPTIONS control statement group syntax consists of the following statements:

```
OPTIONS Statement
INCLUDE Statement
EXCLUDE Statement
END Statement
```

OPTIONS Statement

The OPTIONS statement must be the first statement specified in the control statement group. The OPTIONS statement specifies the maximum number of INCLUDE/EXCLUDE statements that follow. It also specifies whether the inquiries and functions not specified in the INCLUDE/EXCLUDE statements are reloaded (INCLUDE) or not (EXCLUDE).

The format of the OPTIONS statement is:

OPTIONS ENTRIES=25 | nnn),IMODE=(INCLUDE | EXCLUDE),
FMODE=(INCLUDE | EXCLUDE)

The OPTIONS statement parameters are:

ENTRIES=nnn	Specifies the maximum number of INCLUDE and EXCLUDE statements that follow. The default value is 25.
INCLUDE or EXCLUDE	Specifies the disposition of all inquiries and functions that are not covered by INCLUDE and EXCLUDE statements. The default value for both IMODE (inquiries) and FMODE (functions) is INCLUDE. This means that all inquiries and functions not covered by an INCLUDE or EXCLUDE statement are included in the reloaded system database.

INCLUDE / EXCLUDE Statements

If INCLUDE/EXCLUDE statements are not specified, the entire database is reloaded. By specifying INCLUDE/EXCLUDE statements, you can control which inquiries and functions are reloaded.

For instance, you can specify that only inquiries defined by specific logical terminals be included. You can also specify that only those inquiries and functions that begin with a special combination of letters be reloaded. The level of qualification is up to you. These same capabilities pertain to EXCLUDE as well, but in the case of EXCLUDE, inquiries and functions are not reloaded.

The format of the INCLUDE/EXCLUDE statement is:

{ EXCLUDE / INCLUDE } (APPL=applname)(,DIRECTORY=dirname) (TERM=termname)(,TYPE=type)(,NAME=xxx)(,KEY=kkk) (,DATE=yyyymmdd)

The INCLUDE/EXCLUDE statement parameters are:

APPL=applname	Specifies the VISION:Inquiry logical application name to which this statement applies. If omitted, this statement applies to all applications.
DIRECTORY= dirname	Specifies the directory name to which this statement applies. If omitted, this statement applies to all directories within the specified application.
TERM=termname	Specifies the terminal name to which this statement applies. If omitted, this statement applies to all terminals within the specified application.

TYPE=type

Specifies either 'I' or 'F' (for inquiries or functions). If omitted, this statement applies to both inquiries and functions.

NAME=xxx

Specifies the name of an inquiry or function to which this statement applies.

- If omitted, this statement applies to all inquiries or functions (depending on what was specified in the TYPE parameter).
- This parameter is mutually exclusive with the KEY parameter. If both are specified, only the last is used.

KEY=kkk

Specifies the beginning letter pattern of a group of inquiries or functions to which this statement applies. For example, if 'KEY=II13' is specified, this applies to inquiries or functions that begin with 'II13.'

- If omitted, this statement applies to all inquiries and functions (depending on what is specified in the TYPE parameter).
- This parameter is mutually exclusive with the NAME parameter. If both are specified, only the last is used.

DATE=yyyymmdd

Specifies a limiting date for inquiry or function inclusion or exclusion. Inquiries and functions are included or excluded if they were last replaced (or created, if never replaced) before this date. If this parameter is omitted, the last replacement date of stored inquiries and functions is not a factor in inclusion or exclusion.

END Statement

The END statement must be the last statement in the control statement group.

The format of the END statement is:

END

No parameters are required. The END statement must be present even if no other statements are specified.

Sample OPTIONS Control Statement Group

The following is a sample of a typical OPTIONS control statement group:

```
1       OPTIONS ENTRIES=10,IMODE=EXCLUDE,FMODE=EXCLUDE
2       INCLUDE TERM=DY1
3       INCLUDE APPL=II,DIRECTORY=IIDIR,TYPE=F
4       INCLUDE KEY=ABS,TYPE=I
5       INCLUDE NAME=FINDSKILL
6       EXCLUDE APPL=II,DATE=19940725
7       END
```

Figure 1

Statement	Explanation
1	This statement specifies that no more than 10 INCLUDE and EXCLUDE statements follow and that all inquiries and functions that are not specified in an INCLUDE statement are excluded from the reloaded system database.
2	This statement specifies that all inquiries and functions that have been defined by the terminal named 'DY1' are to be included in the reloaded system database.
3	This statement specifies that all functions that have been defined for the directory named 'IIDIR' for the application named 'II' are to be included in the reloaded system database.
4	This statement specifies that all inquiries that begin with the characters 'ABS' are to be included in the reloaded system database.
5	This statement specifies that all inquiries or functions named 'FINDSKILL' are to be included in the reloaded system database.
6	This statement excludes stored inquiries and functions which have not been created or updated for the application named 'II' since before July 25, 1994.
7	This statement is the END statement and it terminates the control statement group.

IXULOAD JCL Requirements

Use one of the following JCL samples, as appropriate to the system database:

- II.TCVLCNTL (IMSLOAD)
- II.TCVLCNTL (DB2LOAD)
- II.TCVLCNTL (VSAMLOAD)

II.TCVLCNTL (IMSLOAD)

Control library member II.TCVLCNTL (IMSLOAD) contains the following sample JCL for allocating, initializing, and reloading an IMS (DL/I) system database.

```
//* IMS SYSTEM DATABASE ALLOCATION AND RELOAD JCL:
//*   -THIS JOB WILL DEFINE AN IMS SYSTEM DATABASE
//*    TO VSAM, PERFORM AN INITIAL LOAD, AND RELOAD
//*    FROM A BACKUP PRODUCED BY IXUUNLD.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS REQUIRED.
//*   - REPLACE ?????? IN THE DEFINE COMMAND WITH THE VOLUME SERIAL
//*     NUMBER OF THE VOLUME ON WHICH THE SYSTEM DATABASE WILL RESIDE.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE AND THE DBDS AND
//*     PSBS GENERATED.
//*
//*
//* DEFINE THE SYSTEM DATABASE CLUSTER TO VSAM
//*
//DEFINE   EXEC PGM=IDCAMS,REGION=500K
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER ( NAME(II.IXXDB) -
                   VOLUMES(??????) -
                   RECORDS(206) -
                   NONINDEXED -
                   RECORDSIZE(2553,2553) -
                   CONTROLINTERVALSIZE(2560) )
/*
//*
//*  INITIALIZE THE SYSTEM DATABASE
//*
//IIINIT   EXEC DLIBATCH,MBR=IIINIT,PSB=IIPSB01
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//IXXDB    DD DISP=OLD,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
//SYSIN    DD *
INDEX=5,DIRECTORY=200;
/*
//*
//*  RELOAD THE SYSTEM DATABASE
//*
//IXULOAD EXEC DLIBATCH,MBR=IXULOAD,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
```

Figure 5-7 Sample JCL to Reload an IMS (DL/I) System Database (Page 1 of 2)

```
//IXXDB    DD DISP=OLD,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
//SYSUT1   DD DISP=SHR,DSN=II.IXXDB.UNLOAD
//SYSIN    DD *
END
/*
```

Figure 5-7 Sample JCL to Reload an IMS (DL/I) System Database (Page 2 of 2)

II.TCVLCNTL (DB2LOAD)

Control library member II.TCVLCNTL (DB2LOAD) contains the following sample JCL for initializing and reloading a DB2 system database.

```
//* DB2 SYSTEM DATABASE RELOAD JCL:
//*   -THIS JOB WILL PERFORM A DB2 SYSTEM DATABASE AN INITIAL
//*    LOAD AND RELOAD FROM A BACKUP PRODUCED BY IXUUNLD.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES IN
//*     THE PARM FIELDS; OR YOU MAY OMIT THE PARMS ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - THE SMP/E INSTALLATION AND INSTALLATION STEPS USING
//*     II.TCVLCNTL MEMBERS DB2BIND AND DB2CREAT MUST BE COMPLETE.
//*
//*
//* INITIALIZE THE SYSTEM DATABASE
//*
//IIINIT   EXEC PGM=IIINIT,REGION=2M,
//         PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//         DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
INDEX=5,DIRECTORY=200;
/*
//*
//* RELOAD THE SYSTEM DATABASE
//*
//IXULOAD EXEC PGM=IXULOAD,REGION=2M,
//         PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//         DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=II.IXXDB.UNLOAD
//SYSIN    DD *
END
/*
```

Figure 5-8 Sample JCL to Reload a DB2 System Database

II.TCVLCNTL (VSAMLOAD)

Control library member II.TCVLCNTL (VSAMLOAD) contains the following sample JCL for allocating, initializing, and reloading a VSAM system database:

```
//* VSAM SYSTEM DATA BASE ALLOCATION AND RELOAD JCL:
//*   - THIS JOB WILL DEFINE A VSAM SYSTEM DATA BASE
//*     TO VSAM, PERFORM AN INITIAL LOAD, AND RELOAD
//*     FROM A BACKUP PRODUCED BY IXUUNLD.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - REPLACE ?????? IN THE DEFINE COMMAND WITH THE VOLUME SERIAL
//*     NUMBER OF THE VOLUME ON WHICH THE SYSTEM DATA BASE WILL RESIDE.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE.
//*   - VERIFY OR CHANGE THE DD NAME "IXXDB" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*
//*
//* DEFINE THE SYSTEM DATA BASE CLUSTER TO VSAM
//*
//DEFINE   EXEC PGM=IDCAMS,REGION=500K
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DEFINE CLUSTER ( NAME(II.IXXDB) -
                   VOLUMES(??????) -
                   RECORDS(206) -
                   NUMBERED -
                   RECORDSIZE(2040,2040) )
/*
//*
//*  INITIALIZE THE SYSTEM DATA BASE
//*
//IIINIT   EXEC PGM=IIINIT,REGION=500K,
//         PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=OLD,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
INDEX=5,DIRECTORY=200;
/*
//*
//*  RELOAD THE SYSTEM DATA BASE
//*
//IXULOAD  EXEC PGM=IXULOAD,REGION=520K,
//         PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=OLD,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=II.IXXDB.UNLOAD
//SYSIN    DD *
END
/*
```

Figure 5-9 Sample JCL for Allocating, Initializing and Reloading a VSAM System Database

Unloading the Stored Inquiries and Functions with IXUSQRY

The IXUSQRY utility creates an unloaded copy of the stored inquiries and functions in a sequential, 80-byte fixed length record data set. This data set is primarily used as input to the batch version of the product to store the inquiries and functions back in the system database when you need to re-create the directories which causes the stored inquiries and functions to be deleted.

IXUSQRY also generates the necessary control statements and commands in the data set which will be used by the batch version. You may unload the stored inquiries and functions using this utility, recreate directories using the IIGEN utility, and restore the inquiries and functions again in the system database using the batch version of the product.

IXUSQRY Statement

The IXUSQRY utility input statements unload the inquiries and functions stored for:

- A specific directory or directories
- A specific map within a directory
- A specific map within an application
- A specific map in the system database
- A specific application or applications
- All applications.

The format of the IXUSQRY statement is:

```
{ APPL=applname (,DIRECTORY=dirname)(,MAP=dbname) }
{ MAP=dbname                                      }
{ ALL                                             }
```

The IXUSQRY statement parameters are:

APPL=applname	Defines the application name to which the utility applies.
DIRECTORY=dirname	Defines the directory name to which the utility applies. If this parameter is specified, the APPL parameter must also be specified.

MAP=dbname	Specifies the database or file name (the NAME parameter of the MAPGEN statement) to which the utility applies. This can be a stand-alone parameter, to unload all the stored inquiries and functions for the database or file in the system database. It can also be used with APPL and DIRECTORY parameters.
ALL	If specified, the utility applies to the whole system database.

Notes:

- At least one control statement must be specified for this utility.
- More than one control statement can be specified for this utility but the parameters cannot be repeated in one statement.
- The END statement must be the last statement in the control statement group. No parameters are required for the END statement.
- This utility will also unload the native SQL syntax stored inquiries. MAP=EXECSQL should be used to unload only the native SQL syntax inquiries.

IXUSQRY Statement Example:

The following is a sample specification of a typical control statement group:

```
1          APPL=IQIO,DIRECTORY=DIR1

2          APPL=IQIO,DIRECTORY=DIR2

3                  MAP=SKILL

4          APPL=IQ11

5          END
```

Statement	Explanation
1	This statement specifies that the stored inquiries and functions in directory named 'DIR1' in the application named 'IQIO' are to be unloaded.
2	This statement specifies that the stored inquiries and functions in directory named 'DIR2' in the application named 'IQIO' are to be unloaded.
3	This statement specifies that all the inquiries and functions stored for the database/file with the name of SKILL in the system database are to be unloaded.
4	This statement specifies that the stored inquiries and functions of all the directories for the application named 'IQ11' are to be unloaded.
5	The END command terminates the control statement group.

IXUSQRY JCL Requirements

Use one of the following JCL samples, as appropriate to the system database:

- II.TCVLCNTL (IMSSQRY)
- II.TCVLCNTL (DB2SQRY)
- II.TCVLCNTL (VSAMSQRY)

II.TCVLCNTL (IMSSQRY)

Control library member II.TCVLCNTL (IMSSQRY) contains the following sample JCL for unloading the source of the stored inquiries and functions from an IMS DL/I system database.

```
//* IMS SYSTEM DATABASE SOURCE INQUIRY/FUNCTION UNLOAD JCL:
//*
//*   - THIS JOB UNLOADS THE SOURCE OF STORED INQUIRIES/FUNCTIONS
//*     FROM THE SYSTEM DATABASE TO A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET MUST HAVE
//*     ENOUGH SPACE FOR ALL THE UNLOADED STORED INQUIRIES/FUNCTIONS.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE AND THE DBDS AND
//*     PSBS GENERATED.
//*   - ADD THE NECESSARY INPUT CONTROL STATEMENTS AS SYSIN DATA.
//*
//*
//IXUSQRY  EXEC DLIBATCH,MBR=IXUSQRY,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.SOURCE.INQUIRY.UNLOAD
//SYSIN    DD *
  Input control statements for IXUSQRY utility
  END
/*
```

Figure 5-10 Sample JCL to Unload Stored Inquiries or Functions from an IMS (DL/I) System Database

II.TCVLCNTL (DB2SQRY)

Control library member II.TCVLCNTL (DB2SQRY) contains the following sample JCL for unloading the source of the stored inquiries and functions from a DB2 system database.

```
//* DB2 SYSTEM DATABASE SOURCE INQUIRY/FUNCTION UNLOAD JCL:
//*
//*   - THIS JOB UNLOADS THE SOURCE OF STORED INQUIRIES/FUNCTIONS
//*     FROM THE SYSTEM DATABASE TO A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES IN
//*     THE PARM FIELDS; OR YOU MAY OMIT THE PARMS ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET MUST HAVE
//*     ENOUGH SPACE FOR ALL THE UNLOADED STORED INQUIRIES/FUNCTIONS.
//*   - ADD THE NECESSARY INPUT CONTROL STATEMENTS AS SYSIN DATA.
//*
//*
//IXUSQRY  EXEC PGM=IXUSQRY,REGION=2M,
//         PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//         DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.SOURCE.INQUIRY.UNLOAD
//SYSIN    DD *
 Input control statements for IXUSQRY utility
 END
/*
```

Figure 5-11 Sample JCL to Unload Stored Inquiries/Functions from a DB2 System Database

II.TCVLCNTL (VSAMSQRY)

Control library member II.TCVLCNTL (VSAMSQRY) contains the following sample JCL for unloading the source of the stored inquiries in a VSAM system database.

```
//* VSAM SYSTEM DATA BASE SOURCE INQUIRY UNLOAD JCL:
//*
//*   - THIS JOB UNLOADS THE SOURCE OF STORED INQUIRIES FROM THE
//*     SYSTEM DATA BASE TO A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DD NAME "IXXDB" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET MUST
//*     HAVE ENOUGH SPACE FOR ALL THE UNLOADED STORED INQUIRIES.
//*   - ADD THE NECESSARY INPUT CONTROL STATEMENTS AS SYSIN DATA.
//*
//*
//IXUSQRY  EXEC PGM=IXUSQRY,REGION=1024K,PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.SOURCE.INQUIRY.UNLOAD
//SYSIN    DD *
  ---  INPUT CONTROL STATEMENTS FOR IXUSQRY UTILITY  ---
  END
/*
```

Figure 5-12 Sample JCL for Unloading Stored Inquiries in a VSAM System Database

Converting Stored Inquiries and Functions with IXUIQRY

Prior to Release 6.0, stored inquiries and functions were kept in internal format only. With the release of 6.0, stored inquiries and functions are kept in two formats: in internal format for execution, and in source format.

- The source format of the stored inquiries is used by the Text Editor for editing.
- Native SQL syntax inquiries are stored in source format only. This is used for execution and by the Text Editor for editing.
- The source format of the stored inquiries and functions can also be unloaded using the IXUSQRY utility in the case of recreating directories.

To upgrade the system database, you need to unload the system database using the unload utility of your existing release, and then reload using the load utility of the new release.

If you are upgrading from a release prior to 6.0, an extra step (IXUIQRY) is required. This extra step creates and saves the source format of the stored inquiries and functions in the system database.

IXUIQRY converts the stored inquiries and functions from internal format to source, and saves them on an 80-byte sequential data set. This data set can then be used as input to the batch version of the product to restore and create the source format of the inquiries and functions in the system database. Note that the necessary control statements and commands for the batch version to store the inquiries and functions back to the system database will also be generated in the sequential data set along with the source of the inquiries and functions.

There are no input statements for this utility.

Notes:

- During the conversion process, the IXUIQRY utility will replace the stored function names in the stored inquiries with their actual expressions.
- If you already have the source of the stored inquiries and functions, we highly recommend that you use it as input to the batch version as opposed to the source created by the IXUIQRY utility. The source created by the IXUIQRY may appear different from the way it looked when the inquiry and function was stored, especially if it is a UDO inquiry.
- The IXUIQRY will not unload the native SQL syntax stored inquiries because there is no internal format for those inquiries in the system database.

IXUIQRY JCL Requirements

Use one of the following JCL samples, as appropriate to the system database:

- II.TCVLCNTL (IMSIQRY)
- II.TCVLCNTL (DB2IQRY)
- II.TCVLCNTL (VSAMIQRY)

II.TCVLCNTL (IMSIQRY)

Control library member II.TCVLCNTL (IMSIQRY) contains the following sample JCL to convert stored inquiries and functions to source format for an IMS DL/I system database.

```
//* IMS SYSTEM DATABASE CONVERSION TO SOURCE INQUIRY/FUNCTION JCL:
//*
//*   - THIS JOB CONVERTS THE STORED INQUIRIES/FUNCTIONS FROM INTERNAL
//*     TO SOURCE FORMAT AND SAVES THEM IN A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE AND THE DBDS AND
//*     PSBS GENERATED.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET MUST HAVE
//*     ENOUGH SPACE FOR ALL THE CONVERTED, STORED INQUIRIES/FUNCTIONS.
//*
//IXUIQRY  EXEC DLIBATCH,MBR=IXUIQRY,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.CONVERT.INQUIRY.UNLOAD
```

Figure 5-13 Sample JCL to Convert Inquiries/Functions for an IMS (DL/I) System Database

II.TCVLCNTL (DB2IQRY)

Control library member II.TCVLCNTL (DB2IQRY) contains the following sample JCL to convert the stored inquiries and functions to source format for a DB2 system database.

```
//* DB2 SYSTEM DATABASE CONVERSION TO SOURCE INQUIRY/FUNCTION JCL:
//*
//*   - THIS JOB CONVERTS THE STORED INQUIRIES/FUNCTIONS FROM INTERNAL
//*     TO SOURCE FORMAT AND SAVES THEM IN A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES IN
//*     THE PARM FIELDS; OR YOU MAY OMIT THE PARMS ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET MUST HAVE
//*     ENOUGH SPACE FOR ALL THE CONVERTED, STORED INQUIRIES/FUNCTIONS.
//*
//IXUIQRY  EXEC PGM=IXUIQRY,REGION=2M,
//         PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//         DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.CONVERT.INQUIRY.UNLOAD
```

Figure 5-14 Sample JCL to Convert Inquiries/Functions for a DB2 System Database

II.TCVLCNTL (VSAMIQRY)

Control library member II.TCVLCNTL (VSAMIQRY) contains the following sample JCL for conversion of the stored inquiries to source format in a VSAM system database.

```
//* VSAM SYSTEM DATA BASE CONVERSION TO SOURCE INQUIRY JCL:
//*
//*   - THIS JOB CONVERTS THE STORED INQUIRIES FROM INTERNAL TO
//*       SOURCE FORMAT AND SAVES THEM IN A SEQUENTIAL DATA SET.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DD NAME "IXXDB" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - VERIFY OR CHANGE THE PARAMETERS FOR SYSUT1 DD STATEMENT.
//*     NOTE THAT LRECL PARAMETER MUST BE 80 AND THE DATA SET
//*     MUST HAVE ENOUGH SPACE FOR ALL THE STORED INQUIRIES.
//*
//*
//IXUIQRY  EXEC PGM=IXUIQRY,REGION=1024K,PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DCB=(RECFM=FB,BLKSIZE=4000,LRECL=80),
//            UNIT=SYSDA,DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(3,1)),
//            DSN=II.CONVERT.INQUIRY.UNLOAD
```

Figure 5-15 Sample JCL for Conversion of Stored Inquiries in a VSAM System Database

IXUIQRY Sample Output

Figure 5-16 shows a sample output data created by the IXUIQRY utility in the sequential data set.

```
/SET TRAN IQIO
/SET IAM LTERM BATCH
/SET SEQ
DDI PLANT 'EMPINFO' DISPLAY PLANT PLANT.ID EMP.NO;;
DDI SKILL 'SKILLINFO' DISPLAY SKILL SKILL.CODE SKILL.NAME;;
```

Figure 5-16 Sample Output Data Created by IXUIQRY

Reporting System Database Statistics with IXUSTAT

The IXUSTAT utility provides information about the components and their space utilization on the system database. This information can be used to determine the optimum organization of applications, maps, and directories to minimize the number of system database accesses made by the inquiry processing programs. By running IXUSTAT periodically, you can also monitor the amount of space being used by defined inquiries, defined functions, and deferred inquiries.

Understanding the Report Contents

The statistics report is produced by accessing all of the index blocks on the system database and individually analyzing each of the entries encountered.

In order to fully utilize the information on this report, an understanding of the basic structure and contents of the system database is required. For information about the system database, see the sample report in Figure 5-17 on page 5-44. Information about defining maps, directories, and terminals is in Chapter 4, "The Definition Process".

IXUSTAT Report Contents

IXUSTAT provides information on the following:

- *System database statistics*
  Provides space utilization statistics.
- *Inquiry and function*
  Provides information about inquiries and functions that are in the system database.
- *Directory*
  Provides the names of the maps in each directory and relationships between primary and connected directories.

Each of these is described in detail.

System Database Statistics Report

Figure 5-17 illustrates a System Database Statistics report. The superscript numbers (1), (2), (3), and so on, are explained in the text following the figure.

COMPONENT	ENTRY NAME	ENTRY COUNT	TOTAL BYTES	AVG BYTES	BLOCK COUNT	FILE/DIR TYPE/KEY
APPL(1)	$$$$$IXX					
INDEX BLOCK NUMBER	0001(11)					
ERROR MESSAGE(3)	IXXERROR	13	26,416	2,032	13	
VOCABULARY	SYSVOCAB	128	2,830	22	2	
NOISE-WORDS(4)		20	420	21		
APPL TOTAL(2)	$$$$$IXX	141	29,246	207	15	
APPL	IQIO					
GENERIC TERM	D*******					61(24)
TERMS		0	0	0	1	
DEFERS		0	0	0	0	
GENERIC TERM(17)	NMSPG0**					55
TERMS(18)		2(15)	40	20	1	
DEFERS(19)		1	0	0	1	
GENERIC TERM	NMSPG1**					55
TERMS		0	0	0	1(16)	
DEFERS		0	0	0	0	
GENERIC TERM	********					55
TERMS		2	40	20	1	
DEFERS		1	0	0	1	
ORDINARY TERM	BATCH(5)					55
DEFERS		1(6)	0	0	1	
ORDINARY TERM	PC					55
DEFERS		1	0	0	1	
MAP	FTSMAP	51	3,128	61	2(10)	IMS
SEGMENTS		3(7)	168	56		
FIELDS		23	586	25(9)		
SYNONYMS		0	0	0		
INDEX FIELDS		0	0	0		
SEARCH FIELDS		0	0	40		
MAP	IIDMMAP	52	3,200	61	2	IMS
SEGMENTS		6	328	54		
FIELDS		22	594	27		
SYNONYMS		0	0	0		
INDEX FIELDS		0	0	0		
SEARCH FIELDS		0	0	0		
MAP	IIDSMAP	23	1,402	60	1	IMS
SEGMENTS		3	160	53		
FIELDS		6	150	25		
SYNONYMS		3	62	20		
INDEX FIELDS		0	0	0		
SEARCH FIELDS		0	0	0		
MAP	VSHPLMAP	69	4,360	63	3	VSMHKSDS
SEGMENTS		7	428	61		
FIELDS		26	760	29		
SYNONYMS		4	126	31		
INDEX FIELDS		0	0	0		
SEARCH FIELDS		0	0	0		
MAP	VSHSKMAP	24	1,538	64	1	VSMHKSDS(20)
SEGMENTS		4	240	60		
FIELDS		7	200	28		
SYNONYMS		2	68	34		
INDEX FIELDS		0	0	0		
SEARCH FIELDS		0	0	0		
MAP	VSPLMAP	19	1,212	63	1	VSAMKSDS
RECORD		56	56			
FIELDS		8	222	27		
SYNONYMS		0	0	0		
INDEX FIELDS		0	0	0		
MAP	VSSKMAP	11	720	65	1	VSAMRRDS
RECORD		56	56			
FIELDS		4	114	28		
SYNONYMS		0	0	0		
INDEX FIELDS		0	0	0		

Figure 5-17 Sample System Database Statistics Report (Page 1 of 2)

```
  DIRECTORY           ALL           241 (8)      5,880          24       3          55
    DATE              1994.077 (21)
    TIME              183059.2 (22)
  MAPS                                  7          924          44
    FIELDS                             96        2,364          24
    SYNONYMS                            9          222          24
  VOCABULARY                          128        2,944          23
    INQUIRIES (25)                      0            0           0
    TEXT INQ (26)                       0            0           0
    FUNCTIONS                           0            0           0
    INDEX FIELDS                        0            0           0
    SEARCH FIELDS                       0            0           0
  DIRECTORY           VSAM          184        4,436          24       3          61 (23)
    DATE              1995.077
    TIME              183053.3
  MAPS                                  4          176          44
    FIELDS                             45        1,112          24
    SYNONYMS                            6          156          26
  VOCABULARY                          128        2,944          23
    INQUIRIES                           0            0           0
    TEXT INQ                            0            0           0
    FUNCTIONS                           0            0           0
    INDEX FIELDS                        0            0           0
    SEARCH FIELDS                       0            0           0
INDEX BLOCK NUMBER    0002 (12)
INDEX BLOCK NUMBER    0003
INDEX BLOCK NUMBER    0004
APPL TOTAL            IQIO          676       25,876          38      23

INDEX (13)                          17
  ERROR MESSAGES                     1
  VOCABULARIES       .               1
  MAPS                               7
  DIRECTORIES                        2
  GENERIC TERMS                      4
  ORDINARY TERMS                     2
  FREE SPACE                                 7,492
FREE DIRECTORY BLKS:                                                      161
```

Figure 5-17 Sample System Database Statistics Report (Page 2 of 2)

The following is an explanation for the column headings and the numbers shown in the System Database Statistics report above.

COMPONENT

Listed below the heading COMPONENT are the component types for which space statistics are produced. Note that some of the components are indented.

As described in System Database Index on page 3-3, index entries are maintained in sequence by application name, entry type, and entry name. Note that APPL and APPL TOTAL are not indented. All components that appear between these lines are for the application name appearing under the heading ENTRY NAME. For example, at (1), $$$$$IXX is indicated as an application name. All of the indented items down to APPL TOTAL at (2) are for the application name $$$$$IXX.

All of the components that are indented two columns involve an index entry.

- For example, at (3), the index entry for the system error message is indicated.
- At (17), the statistics for a generic terminal index listing are shown.
- Specifically at (18) is the number of directory blocks and entries used to store the names of the specific TERMs, that have been permitted access to VISION:Inquiry because they match a generic TERM.
- At (19), these statistics are the total of entries, bytes, average bytes, and directory blocks used for inquires deferred from a TERM that gained access to VISION:Inquiry because it matched a generic TERM.

The name under the heading ENTRY NAME plus the application make up the index entry name that is used when searching the index.

All of the components that are indented four columns are actually sub-components of the component they follow. Note that they have no entry name which means they do not have an entry in the index.

- For example, at (4), there is a sub-component for the number of noise words in the standard system vocabulary.
- The sub-component INQUIRIES at (25) gives the statistics about the internal format of the stored inquiries.
- The sub-component TEXT INQ at (26) gives the statistics about the source format of the stored inquiries used by the Text Editor.

ENTRY NAME

Listed below the heading ENTRY NAME are the entry names in the index for the different components. The actual index entry name is made up of the application name plus the entry name for the component involved. For example, at (5), the index entry name for the terminal BATCH is made up of the application name II and the component name BATCH.

In addition, for each directory under this heading appear the date, YYYY.DDD (Julian), and the time, HHMMSS.T, that the directory was last updated. This is illustrated at (21) and (22).

ENTRY COUNT

Listed below the heading ENTRY COUNT is a count of the indicated components in the entry. This count may be the number of deferred inquiries for a terminal (6), the number of segments in a map for a database (7), the total number of sub-components (8), or the number of specific TERM names stored for this generic TERM (15).

TOTAL BYTES

Listed below the heading TOTAL BYTES is the total number of bytes required to store the indicated component or components.

AVG BYTES

Listed below the heading AVG BYTES is the average number of bytes required to store the component or sub-components. For example, at (9), the average number of bytes required to store a field definition for this map is 25. This is obtained by dividing the TOTAL BYTES by the ENTRY COUNT.

BLOCK COUNT

Listed below the heading BLOCK COUNT is the number of blocks required for this component.

- For example, at (10), the number of blocks required is 2.
- At (16) the number of blocks used to store the specific TERM entries for the generic terminal is listed.

As the statistics report is being produced, information on the index is interleaved with the basic space information. Each time a different index block is read, a line is printed indicating the index block number.

- For example, at (11), it is indicated that index block 0001 was read. All of the components that follow have their entries in index block 0001.
- At (12), an indication that the next index block was read appears.

As you can see, there are no entries in index blocks 0002, 0003, or 0004.

Statistics about the index are given at (13). The INDEX line is the total number of possible index entries. The following lines indicate how these entries are used.

FILE TYPE/DIR KEY

When the heading FILE TYPE/DIR KEY is used for map entries, it lists the type of file for which this map is designed. For example, at (20), the map VSHSKMAP is defined as a hierarchical VSAM KSDS file.

When this heading is used for a directory entry, it lists the system database entry key where the first record is defined. When this heading is used for a generic or ordinary terminal entry, it lists the associated key of the directory with which it is associated. For example, the generic terminal D******* at (24) is associated with the directory VSAM at (23).

Inquiry and Function Report

This report lists all of the inquiries and functions on the system database. The report includes the following:

- Inquiry or function name
- Application and directory where it resides
- Terminal from which the inquiry or function was defined
- Date and time it was defined
- Terminal from which the inquiry or function was modified
- Last date and time it was modified
- Number of bytes required to store it in internal and source format

If no modifications were made, the entries under the word MODIFIED contain asterisks. Figure 5-18 shows a sample Inquiry and Function Report.

```
                                                 ————————CREATED————————  ————————  MODIFIED  ————————INT     SRC
TYPE INQUIRY-OR-FUNCTION-NAME-APPL--DIRECT-TERMINAL-DATE  ———  TIME—— TERMINAL-DATE———    TIME—BYTES — BYTES

INQY BONUS.PLUS               IQIO  ALL    BATCH    2002-05-28 14:46  DY3      2002-05-30 16:33  200     350
INQY BONUS.SKILL              IQIO  ALL    BATCH    2002-05-28 14:46  ******** ******   ****     748     500
INQY CONV.EXIT                IQIO  ALL    BATCH    2002-05-28 14:46  DY3      2002-05-30 16:35  200     220
INQY EMPLOYEE.LISTING         IQIO  ALL    BATCH    2002-05-28 14:46  *******  ******   ****     296     190
INQY EMPLOYEE.SALARY          IQIO  ALL    BATCH    2002-05-28 14:46  *******  ******   ****     928     462
FUNC MON.SAL                  IQIO  ALL    BATCH    2002-05-28 14:46  *******  ******   ****     244     201
INQY SAL.DATA                 IQIO  ALL    BATCH    2002-05-28 14:46  *******  ******   ****     244     484
FUNC YEARLY.BONUS             IQIO  ALL    BATCH    2002-05-28 14:46  DY2      2002-06-02 11:11  220     284
```

Figure 5-18 Sample Inquiry and Function Report

Directory Report

This report gives detailed information about the directories in the system database. This information is particularly helpful when updating maps in a directory. The report consists of two types of information:

- The map names defined in each directory in the order stored in the directory.
- For connected directories, the relationship between each directory with its connected directories.

Figure 5-19 shows a sample directory report.

```
COMPONENT------------ ENTRY-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME
MAP-NAME

APPL.............   IQIO
  DIRECTORY.........   ALL
    MAPS.............             IIDMMAP   IIDSMAP   FTSMAP
  DIRECTORY.........   IIDMDIR
    MAPS.............             **** NONE FOUND.
  DIRECTORY.........   VSAM
    MAPS.............             VSPLMAP  VSSKMAP   VSHPLMAP  VSSKMAP


     PRIMARY                                    CONNECTED
APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--

IQIO    /ALL

IQIO    /IIDMDIR  ===> IQIO    /ALL

IQIO    /VSAM     ===> IQIO    /IIDMDIR  ===> IQIO    /ALL
```

Figure 5-19 Sample Directory Report

Page 1 of the report in Figure 5-19 shows that the system database contains one application (IQIO) with three directories (ALL, IIDMDIR, and VSAM). Following each directory the map names defined in that directory are shown. For example, three maps (IIDMMAP, IIDSMAP, and FTSMAP) are defined in the directory ALL.

Page 2 of the report shows the relationship between directories: directory ALL is not connected to any other directory, directory IIDMDIR is connected to directory ALL, and directory VSAM is connected to directory IIDMDIR (which in turn connected to directory ALL).

Using the Report Information to Improve Performance

In addition to using these reports to keep track of the amount of system database space being used, you can look for other things that can improve the performance of VISION:Inquiry. For example, look for ways to reduce the number of system database calls required to process an inquiry. Some of the things to look for are the following:

- User directories that use a large number of directory blocks.
  User directories should be made as small as possible without sacrificing convenience for the end user. Smaller user directories require fewer calls to translate an inquiry.

  Try to organize your user directories so that a given user directory has only those database maps that are frequently used in a single inquiry that accesses multiple databases with the FIND statement. If there is a possibility of access to the non-frequent databases by the user, the situation can be alleviated by connected directories.

  The search time for the few times the connected directory must be used is offset by the faster access obtained from specifying a smaller primary directory (performance may be affected by specifying a large directory).
- Large numbers of stored inquiries and functions in a user directory.
  Determine whether your users access ad hoc inquiries or stored inquiries. If they use both, determine if one is used more often than the other. If this is the case, set up two different applications with a directory in each. In one directory include the maps but remove the DEFINE statement. The directories must be connected to each other. Then, to define or use a stored inquiry, your user must use the application with the directory that contains the DEFINE statement. For ad hoc inquiries, the other application must be used.

IXUSTAT JCL Requirements

Use one of the following JCL members, as appropriate to the system database.

- II.TCVLCNTL (IMSSTAT)
- II.TCVLCNTL (DB2STAT)
- II.TCVLCNTL (VSAMSTAT)

II.TCVLCNTL (IMSSTAT)

Control library member II.TCVLCNTL (IMSSTAT) contains the following sample JCL for generating reports from the IMS (DL/I) system database.

```
//* IMS SYSTEM DATABASE STATISTICS REPORT JCL:
//*   - THIS JOB WILL PRODUCE A REPORT ON THE CONTENTS OF AN
//*     IMS SYSTEM DATABASE.
//*
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE AND THE DBDS AND
//*     PSBS GENERATED.
//*
//*
//IXUSTAT  EXEC DLIBATCH,MBR=IXUSTAT,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCVLPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//DFSVSAMP DD *
2048,6
4096,6
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DRPRINT  DD SYSOUT=*
//IFPRINT  DD SYSOUT=*
```

Figure 5-20 Sample JCL for Generating Reports from the IMS (DL/I) System Database

II.TCVLCNTL (DB2STAT)

Control library member II.TCVLCNTL (DB2STAT) contains the following sample JCL for generating reports from the DB2 system database.

```
//* DB2 SYSTEM DATABASE STATISTICS REPORT JCL:
//*   - THIS JOB WILL PRODUCE A REPORT ON THE CONTENTS OF A
//*     DB2 SYSTEM DATABASE.
//*
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES IN
//*     THE PARM FIELDS; OR YOU MAY OMIT THE PARMS ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - VERIFY OR CHANGE THE DB2 SUB-SYSTEM, PLAN, AND TABLE NAMES IN
//*     THE PARM FIELD; OR YOU MAY OMIT THE PARM ENTIRELY IF THIS
//*     INFORMATION IS SPECIFIED AT SMP/E INSTALLATION TIME.
//*
//*
//IXUSTAT  EXEC PGM=IXUSTAT,REGION=2M,
           PARM='TYPE=DB2,SSID=DSN,PLAN=II,NAME=DYLINQ.IISYSTEM'
//STEPLIB   DD DISP=SHR,DSN=II.TCVLPGM
//          DD DISP=SHR,DSN=DSN510.SDSNLOAD
//SYSPRINT  DD SYSOUT=*
//DRPRINT   DD SYSOUT=*
//IFPRINT   DD SYSOUT=*
```

Figure 5-21 Sample JCL for Generating Reports from the DB2 System Database

II.TCVLCNTL (VSAMSTAT)

Control library member II.TCVLCNTL (VSAMSTAT) contains the following sample JCL for generating reports from a VSAM system database:

```
//* VSAM SYSTEM DATA BASE STATISTICS REPORT JCL:
//*   - THIS JOB WILL PRODUCE A REPORT ON THE CONTENTS OF A
//*     VSAM SYSTEM DATA BASE.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE DD NAME "IXXDB" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*
//*
//IXUSTAT  EXEC PGM=IXUSTAT,REGION=500K,PARM='TYPE=VSAM'
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//IXXDB    DD DISP=SHR,DSN=II.IXXDB
//SYSPRINT DD SYSOUT=*
//DRPRINT  DD SYSOUT=*
//IFPRINT  DD SYSOUT=*
```

Figure 5-22 Sample JCL for Generating Reports from a VSAM System Database

Initializing the VISION:Journey VSAM Download Data Set with VSMFTSIN

The VISION:Journey VSAM download data set is initialized by creating a dummy root record. The VSMFTSIN utility creates the dummy record and initializes the subsequence number to 0.

The control library member II.TCVLCNTL (VSMFTSIN) contains the sample JCL for initializing the VISION:Journey download data set. You can find a listing of this member in the defining and initializing the VISION:Journey Download Data Set in the *Advantage VISION:Inquiry for CICS Installation Guide*.

Maintaining the VISION:Journey Download Data Set with IFUCLEN

The IFUCLEN utility deletes the extracted data from the download data set based on the parameters specified in the OPTIONS control statements. The OPTIONS control statement conforms to the same coding conventions as the other utilities discussed at the beginning of this chapter.

The utility also creates a report listing of all the records deleted.

OPTIONS Control Statement Group

When the extracted data is written to the download data set by VISION:Inquiry, in addition to a unique subsequence number assigned to it for each user, the logical application and terminal name and the extraction date are also saved for each set of data. By specifying the parameters in the OPTIONS control statements, you can control one or more sets of extracted data to be deleted from the download data set.

```
OPTIONS statement
        .
        .
        .
OPTIONS statement
END
```

You can have as many OPTIONS control statements as you need.

OPTIONS Statement

The format of the OPTIONS statement is:

OPTIONS (APPL=applname)(,STAT=x(,x)...)
(,LTERM=termname
(,KEY=num(,num) ...)
(,KEYRANGE=((numr$_1$)(,numr$_2$))(,((numr$_1$)(numr$_2$)))...))
(,DATE=yyyymmdd(,yyyymmdd)...)
(,DATERANGE=((dater$_1$)(,dater$_2$))(,((dater$_1$)(,dater$_2$)))...)

The OPTIONS statement parameters are:

APPL= applname — Specifies the logical application name to which the deletion applies. If omitted, no application checking is done for the deletion.

STAT=x,x — Specifies the status of extracted data to be deleted. It can be I, C, F, or a combination of these, separated by a comma.

- 'I' means that extraction is incomplete and has been stopped abnormally.
- 'C' means that the extraction has reached the checkpoint limit, but not completed.
- 'F' means that data has been extracted completely by VISION:Inquiry, but has not been downloaded yet.
- If status is omitted, the status is not checked for deletion.

LTERM= termname

Specifies the terminal name to which the deletion applies. If omitted, no terminal name checking is done for deletion.

KEY=num

Specifies the unique subsequence numbers to which the deletion applies.

- If omitted, the subsequence number is not used for deletion.
- If this parameter is specified, the LTERM parameter must also be specified.

KEYRANGE= ($numr_1$,$numr_2$)

Specifies the range of subsequence numbers to which the deletion applies.

- If the lower range is omitted, the range begins with the lowest number that exists; you should code a comma to indicate omission.
- If the upper range is omitted, the range ends with the highest number that exists.
- If the KEYRANGE parameter is omitted, no range is used for deletion. If this parameter is specified, the LTERM parameter must also be specified.

DATE= yyyymmdd

Specifies the extraction dates to which the deletion applies. If omitted, no date is checked during deletion.

DATERANGE= ($dater_1$,$dater_2$)

Specifies the range of extraction dates to which the deletion applies.

- If the lower range is omitted, the range begins with the oldest date that exists; you should code a comma to indicate omission.
- If the upper range is omitted, the range ends with the most recent date that exists.
- If the DATERANGE parameter is omitted, no DATERANGE is checked during deletion.

The arguments $dater_1$ and $dater_2$ must be in the "yyyymmdd" format.

Notes:

- At least one of the parameters should be specified.
- The parameters can be in any order and combination, except when KEY and KEYRANGE are specified. If KEY or KEYRANGE is specified, the parameter LTERM must also be specified.
- If KEY and KEYRANGE are specified in an OPTIONS statement, the deletion applies to both parameters; that is, the data associated with both KEY and KEYRANGE will be deleted. The same rule applies to DATE and DATERANGE.
- If more than one parameter is specified in an OPTIONS statement, all parameters must match before deletion applies.
- Master control records (with the subsequence number of zero) cannot be deleted using this utility. Initializing the download database will reset the master control record.

END Statement

The END statement must be the last statement in the control statement group.

The format of the END statements:

END

No parameters are required.

OPTIONS Control Statement Group Sample

The following is a sample specification of a typical OPTIONS control statement group.

```
1       OPTIONS APPL=IQIO,STAT=I,C,LTERM=L001
2       OPTIONS APPL=IQIO,KEYRANGE=(10,70),KEY=120,130,LTERM=L002
3       OPTIONS LTERM=L002,DATERANGE=(,19941231)
4       OPTIONS APPL=IQIO,LTERM=L002,KEY=20
5       END
```

Statement	Explanation
1	Deletes all data extracted by terminal L001 for application IQIO; it has the status of either incomplete or has reached a checkpoint limit.
2	Deletes the extracted data whose subsequence numbers are 120 or 130 or in the range of 10 through 70 if they are created by terminal L002 for application IQIO.
3	Deletes all data extracted by terminal L002 before the end of the year 1994.
4	Deletes the extracted data with subsequence number of 20 if it is created by application IQIO and LTERM L002.
5	The END statement terminates the control statement group.

IFUCLEN JCL Requirements

For a VISION:Journey VSAM download data set, the IFUCLEN utility runs as an ordinary batch job. The control library member II.TCVLCNTL (VSMFTSCL) contains the following sample JCL to execute the cleanup utility for a VISION:Journey VSAM download data set.

The data set name should reference the one that you want to use; the ddname must match that name specified at SMP/E installation time.

The FTSDEL DD statement refers to the listing which gives a report of the records deleted. The report contains information (such as the subsequence number, terminal name, extraction date (in the form YYYYDDD), and so on) of the records deleted.

```
//* VISION:JOURNEY/FTS FEATURE VSAM DOWNLOAD DATA SET CLEANUP JCL:
//*   - THIS JOB WILL DELETE THE DATA/REPORT FROM THE DOWNLOAD
//*     DATA SET. THE PARAMETERS IN OPTIONS CONTROL STATEMENT
//*     SPECIFY WHICH DATA SHOULD BE DELETED.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS REQUIRED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE.
//*   - VERIFY OR CHANGE THE DD NAME "VSFTSDS" TO MATCH WITH
//*     THE NAME SPECIFIED AT SMP/E INSTALLATION TIME.
//*   - CHANGE AND VERIFY THE PARAMETERS OF THE OPTIONS CONTROL
//*     STATEMENT TO REFLECT THE DATA/REPORT WHICH SHOULD BE DELETED.
//*
//*
//*  CLEANUP THE VSAM DOWNLOAD DATA SET
//*
//IFUCLEN  EXEC PGM=IFUCLEN,REGION=1000K
//STEPLIB  DD DISP=SHR,DSN=II.TCVLPGM
//VSFTSDS  DD DSN=VS.VSFTSDS,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FTSDEL   DD SYSOUT=*
//SYSIN DD *
OPTIONS APPL=XXXXXXXX,LTERM=XXXXXXXX,STAT=X,
         KEY=XXXXX,DATE=XXXXXXXX,
         KEYRANGE=(XXXXX,XXXXX),
         DATERANGE=(XXXXXXXX,XXXXXXXX)
 END
/*
```

Figure 5-23 Sample JCL to Clean up the VISION:Journey VSAM Download Data set

The data set name should reference the one that you want to use; the ddname must match that specified at VISION:Inquiry SMP/E installation time.

The FTSDEL DD statement refers to the listing which gives a report of the records deleted. The report contains information (such as the subsequence number, terminal name, extraction date (in the form YYYYDDD), and so on) of the records deleted.

Reorganize the VISION:Journey VSAM Download Data Set with VSMFTSRE JCL

This JCL reorganizes the VISION:Journey VSAM download data set. It eliminates the CI/CA splits and restores unused space to the disk pack. Failure to run this utility affects the performance of VISION:Journey.

II.TCVLCNTL (VSMFTSRE) contains the following JCL which reorganizes the VSAM download file. Schedule this JCL when CICS is down or the download file is not used in the online environment.

```
//* REORGANIZE JCL FOR VISION:JOURNEY/FTS FEATURE VSAM
//* DOWNLOAD DATASET
//*
//*   - THIS PROGRAM ELIMINATES CI/CA SPLITS AND RESTORES
//*     UNUSED SPACE TO THE DISK PACK AND SHOULD BE
//*     SCHEDULED WHEN THE CICS SYSTEM IS DOWN.
//*
//*   - FAILURE TO RUN THIS PROGRAM AFFECTS THE PERFORMANCE
//*     OF VISION:JOURNEY/FTS FEATURE.
//*
//*   - IF YOU RUN THIS JOB MORE THAN ONCE, FOR VS.VSFTSDS:
//*     YOU NEED TO DELETE VS.VSFTSDS.TEMP, OR CHANGE THE JCL
//*     TO DEFINE VS.VSFTSDS.TEMP AS A GDG DATASET.
//*     FOR GDG:
//*     YOU NEED A DSNAME SUBPARAMETER ON THE DCB,
//*     REPRESENTING THE DATASET LABEL.
//*
//* BEFORE SUBMITTING THIS JOB :
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS NEEDED.
//*   - VERIFY OR CHANGE THE SIZE OF THE FILES AS NEEDED.
//*   - VERIFY OR CHANGE UNIT PARAMETER AS NEEDED.
//*   - REPLACE ?????? IN THE DEFINE COMMAND WITH THE VOLUME SERIAL
//*     NUMBER OF THE VOLUME ON WHICH THE DOWNLOAD DATASET WILL RESIDE.
//*
//*
//REORG   EXEC PGM=IDCAMS,REGION=1024K
//SYSPRINT DD  SYSOUT=*
//INDD     DD  DSN=VS.VSFTSDS,DISP=SHR
//OUTDD    DD  DSN=VS.VSFTSDS.TEMP,DISP=(,CATLG),
//         SPACE=(CYL,(5,2)),UNIT=SYSDA,
//         DCB=(RECFM=VB,LRECL=8024,BLKSIZE=8028)
//*
//SYSIN    DD  *
   REPRO  INFILE(INDD) -
          OUTFILE(OUTDD)
  IF LASTCC=0 THEN DO
  DELETE VS.VSFTSDS CLUSTER
  DEFINE CLUSTER (NAME(VS.VSFTSDS)            -
                  SPEED                       -
                  VOLUMES(??????)             -
                  INDEXED)                    -
         DATA (   KEYS(16 0)                  -
                  RECORDSIZE(80 8020)         -
                  RECORDS(50 5) )
  END
  IF LASTCC=0 THEN DO
   REPRO  INFILE(OUTDD) -
          OUTFILE(INDD)
  END
//*
```

Figure 5-24 Sample JCL to Reorganize the VISION:Journey VSAM Data Set

6 Programming and Operation Considerations

VISION:Inquiry can execute in two different environments. A different inquiry processing program is invoked depending on the environment in which it is executed.

Environments and Environment Processors

The following list illustrates the environments and the VISION:Inquiry environment processor programs.

Environment	Inquiry Processor
CICS Online	INQIO Native VISION:Inquiry front-end
CICS Online	IXXSTRT Native VISION:Inquiry main
CICS Online	INQEDIT VISION:Inquiry Text Editor
CICS Online	IDBI01 VISION:Inquiry AQF
CICS Online	INQINTRA Intraccess
CICS Online	DYC0SS, DYLC010, DYLC020, DYLC030 VISION:Journey
Batch, DL/I Batch	IIBATCH Native VISION:Inquiry

This chapter explains:

- The environment-specific inquiry processing programs and guidelines for each operational environment
- Operational considerations for the use of the SORT and EXTRACT commands
- Operational considerations for AQF (Automatic Query Facility) in the online environment:
  - AQF screens (menus) consideration
  - AQF temporary storage queues
  - AQF batch job submission consideration
- Operations considerations for the Intraccess option
- The VISION:Inquiry file transfer operation and the VISION:Journey download data set
- Operational considerations and limitations for the Native SQL syntax facility.

Before reading this chapter, read Chapter 2, “System Overview”.

Additional Load Modules

In addition to the VISION:Inquiry main processors, there are some independent load modules that can be loaded and executed in the VISION:Inquiry process as shown in the table below. The source of these modules, except for DIOSQLC, are available in the source library and can be modified. The source member should then be assembled and linked as described in the *Advantage VISION:Inquiry for CICS Installation Guide*, Chapter 2 "Installation, " in Step 25 - Customize the Parameter and message Modules (Optional). Note that the load module names cannot be changed.

Load Module Name	Description	Environment
CUYGCHK	Changes Checkpoint Intervals in the IIGEN Utility	Batch
CUYSHMG	VISION:Inquiry Base Hard-Coded Messages	Batch, online
CUYSHDG	Utility Programs Report Headings	Batch
CUYXEMSG	Text Editor Hard-Coded Messages	Online
CVLCAJPR	AQF Job Submission parameters	Online
CVLCAMOD	AQF BMS Mapset names	Online
CVLCPARM	CICS parameters module	Online
DIOSQLC	DB2 option independent load module	Batch, online

Program Entry Names for the Batch Version

The native inquiry batch program can be generated with multiple entry points, as described in the following section.

- For CALL Attach, the DB2 subsystem ID and the plan name
- The type of system database (IMS (DL/I), DB2, or VSAM)
- For a DB2 system database, the authorization ID and table name
- For a VSAM system database, the ddname and the password

The names of the inquiry programs, given in Environments and Environment Processors on page 6-1, are member names in the VISION:Inquiry program library built during installation. The program modules are usually copied to other libraries for execution, where they can be renamed; the alternate entry points can be used. Throughout this chapter, the inquiry programs are referred to by their standard names.

Identification Modules

The information VISION:Inquiry needs to connect to DB2, to access the system database, and the Journey support is contained in the identification modules. The following table lists the names of these modules and the program that use them.

ID Module Name	Used By	Program Notes/Conditions
IXBIDENT	IIBATCH	VISION:Inquiry program in batch environment
IXSIDENT	Utilities	When not run under DL/I
	IXXSTRT	VISION:Inquiry main program in CICS (online environment)
	IIBATCH	If IXBIDENT is not built
	IDBI01	Automatic Query Facility (AQF)

The IXSIDENT module can contain only one set of specifications that applies to the utilities. The native CICS online program, IXXSTRT, and the batch inquiry processing, IIBATCH, for which the IXBIDENT identification module has not been built. However, the IXBIDENT inquiry processor identification module can contain up to 16 sets of specifications, each of which creates an entry point to the program. This allows a certain degree of control to be achieved by varying the name (entry point) by which the batch inquiry processing program is called.

Note: If you specify an option in the identification module that you have not purchased, it is not flagged as an error but the program will fail at run time.

System Identification Module, IXSIDENT, and VISION:Inquiry Utilities/Programs

When the system database utility programs (IIINIT, IIGEN, IXUIQRY, IXULOAD, IXUSQRY, IXUSTAT, and IXUUNLD) are executed, the JCL can be used to complete or override the specifications contained in IXSIDENT.

- When the utilities are executed with IMS (using the IMS DLIBATCH procedure), the system database is the IMS (DL/I) database specified in the PSB.
- When IMS is not used, system database information and DB2 connection options can be passed to the utility programs using the PARM field of the JCL EXEC statement. The parameters to be changed are coded in the same way as in the operand field of the IIIDENT macro (see Specifying the IIIDENT Macro Options for VISION:Journey on page 6-8).

If the identification module for the batch inquiry processing program is not built, the program is given a single entry point and the specifications in IXSIDENT are used when it executes; however, the IMS Batch inquiry programs ignore the DB2 connection specifications in IXSIDENT and use IMS Attach to connect to DB2 and the CICS online Inquiry programs ignore the DB2 connection specifications on IXSIDENT and use CICS Attach to connect to DB2.

The system identification module, IXSIDENT, is built and included in the VISION:Inquiry programs as an SMP/E USERMOD during the installation of the VISION:Inquiry, and if you have a DB2 option and/or Journey feature in your system, see the *Advantage VISION:Inquiry for CICS Installation Guide*, Chapter 2 "Installation," Step 11 - RECEIVE a USERMOD for the DB2 option and/or the Journey feature to the Global Zone. The default identification module specifies no DB2, no Journey feature, and IMS system database type.

To apply the identification modules to the system, see Applying the Identification module(s) to the system on page 6-10.

The following sections describe the options concerning the identification modules. You should use the option that best meets your needs. If none of the options given seem appropriate for your installation, you may require a user exit. See Chapter 7, "User Exits" for details about how the input exit can set this information.

Building the System Identification Module IXSIDENT

When the system identification module IXSIDENT is built and the batch identification module is not used, all VISION:Inquiry programs execute according to its specifications. The only exceptions are:

- The IMS Batch inquiry programs ignore the DB2 connection specifications in IXSIDENT, and use IMS Attach to connect to DB2.
- The CICS online Inquiry programs ignore the DB2 connection specifications on IXSIDENT and use CICS Attach to connect to DB2.
- When the utility programs are run under control of IMS (IBM IMS programs and procedures such as DFSRRC00 and DLIBATCH), they always access an IMS (DL/I) system database and ignore the system database type entry in IXSIDENT.

The source library member II.TCVLSRC (IXSIDENT) contains a model for specifying the IIIDENT macro.

To build the system identification module, choose one of the available options:

IIIDENT TYPE=IMS

The system database is an IMS (DL/I) database, determined from the PSB as described above. The connection to DB2 is made from the IMS Batch program using IMS Attach and from the CICS online programs using CICS Attach.

IIIDENT SSID=ssid,PLAN=plan,TYPE=DB2,NAME=authid.tablename

The system database is the DB2 table named authid.tablename. The connection to DB2 is made from the IMS Batch program using IMS Attach or from the ordinary batch program using CALL Attach with subsystem ID “ssid” and plan name “plan.” The connection to DB2 is made from CICS online programs using CICS Attach.

IIIDENT SSID=ssid,PLAN=plan,TYPE=VSAM,NAME=ddname (,LOCK=password)

The system database is the VSAM data set allocated to ddname. If LOCK=password is coded, it specifies the one to eight character VSAM password used to access the system database from all programs except CICS inquiry, for which the password must be specified in the relevant FILE entry.

In the above examples, the connection to DB2 is made from the CICS program using CICS Attach and the connection specifications specified on IXSIDENT is ignored. For the ordinary batch program, the connection is made using CALL Attach with subsystem ID, ssid, and plan name, plan. For the IMS Batch program the connection to DB2 is made using IMS Attach and the connection specifications specified on IXSIDENT is ignored.

Specifying the IIIDENT Macro Options for VISION:Journey

For VISION:Journey, the following options are available in IIIDENT macro:

IIIDENT FTS=x,FTSTR=tranid,FTSMB=bmsname,FTSTY=type, FTSNM=ftsdd, FTSLK=ftspass

Where

FTS=x	Specify Y when you need the support of VISION:Journey; specify N when you do not need the support. If omitted, the default is N. In this case, all other VISION:Journey parameters are ignored.
FTSTR= tranid	Specify one to four character transaction identifier of the first VISION:Journey program. Note that the first VISION:Journey transaction identifier must end with 1. If omitted, the default is FTS1.
FTSMB= bmsname	Define the default BMS mapset name (up to seven characters in length) used in VISION:Journey processing. This screen appears after extraction and before download of data. If omitted, the default is IDFTSP7. VISION:Journey automatically starts the download process after displaying this screen.
FTSTY= type	Specify the download data set type. The default type is IMS. For VISION:Journey for Windows, the type must always be specified as VSAM.
FTSNM= ftsdd	Define the ddname of the VISION:Journey download data set when its type is VSAM. For VISION:Journey, the ddname of the VSAM download data set also is specified at the Installation time.
FTSLK= ftspass	Specify the one to eight character VSAM password for the VISION:Journey VSAM download data set. If it is coded, it specifies the VSAM password to access the VISION:Journey VSAM download data set from the VISION:Journey cleanup batch utility program (IFUCLEN). For the online programs, the password must be specified in the relevant CICS File definition entry.

If the password contains special characters, enclose it in single quotation marks; to include an ampersand or single quotation mark, code two in succession. For example, code:

```
LOCK='&&AB''XY'
```

to specify:

```
&AB'XY
```

as the password. The specified password is stored within the VISION:Inquiry load module in an encrypted form. If more security is required, a security package such as RACF or ACF2 should be used to protect the data set.

The assembly listing for the identification module will include a message giving the value of the encrypted password in hexadecimal notation. When using the PARM field of the EXEC JCL statement to specify a password protected VSAM system database to one of the system database utility programs, use this lock value (not the actual password) in the LOCK operand.

Building the Identification Module IXBIDENT

When the identification module IXBIDENT (batch) is built and applied to the system, the specifications take precedence over IXSIDENT for the inquiry batch program.

The following conditions apply:

- The utility programs use the IXSIDENT specifications by default, but these may be overridden by the JCL.
- IXSIDENT is coded, applied to the system, and interpreted in the same way as described above except for the batch inquiry program for which the batch identification module has been applied.

The IXBIDENT identification module uses the IIIDENT macro. The differences from IXSIDENT are:

- From one to sixteen IIIDENT macros can be specified. Each defines an entry point into the program.
- Each macro must have a unique name starting in column 1.
- The specifications for the first macro apply when the program is entered at its main entry point (when it is invoked by the load module name). The remaining macros, if any, apply when the program is entered at an alternate entry point.

- The name field of the first macro must be IIBATCH. This name is the name of the main load module that must be present in the program library.
- The name fields of the macros other than the first one becomes an alias name in the program library by which the program is invoked to use the specifications on that macro. Therefore, the name used to execute the program determines the set of specifications to be used.

Applying the Identification module(s) to the system

The Identification Modules must be incorporated to the system using the SMP/E facility in the form of USERMOD. The following is an example of an IXBIDENT module with the necessary SMP/E control statements for incorporating into the VISION:Inquiry system.

```
++USERMOD(IQOID01)
  /* USER MOD FOR IXBIDENT IDENTIFICATION MODULE.
  .
++VER(Z038)
  FMID(CCVL650)
  .
++SRC(IXBIDENT) DISTLIB(INQSRCD)
  .
IXBIDENT TITLE 'SYSTEM DATA BASE AND DB2 INFORMATION - BATCH'
*
* NOTE:  ALL THE FOLLOWING REFERENCES TO DB2 APPLIES TO THE DB2 OPTION
*        OF THE PRODUCT ONLY.
*
* THIS MEMBER SPECIFIES THE SYSTEM DATA BASE AND DB2 INFORMATION
* USED BY THE BATCH INQUIRY PROGRAM.  YOU WILL USUALLY USE THIS
* MEMBER ONLY IF YOU USE MORE THAN ONE SYSTEM DATA BASE, AND AT
* LEAST ONE IS NOT AN IMS DATA BASE, OR YOU USE MULTIPLE DB2
* SUB-SYSTEMS.
*
* WHEN USING DB2 DATA BASES THE SUB-SYSTEM ID OF THE DB2 SYSTEM AND
* THE NAME OF THE PLAN BOUND FOR USE BY VISION:INQUIRY ARE SPECIFIED
* BY THE SSID AND PLAN OPERANDS.
*
* YOU MAY SPECIFY FROM ONE TO SIXTEEN IIIDENT MACROS.  EACH DESIGNATES
* AN ENTRY POINT INTO THE LOAD MODULE IIBATCH, A UNIQUE NAME FOR WHICH
* MUST BE SPECIFIED IN THE NAME FIELD (COLUMN 1).  THE FIRST WILL BE
* THE MAIN ENTRY POINT OF THE MODULE MUST BE IIBATCH.
* THE REMAINING ENTRIES WILL BE ALIAS NAMES.
* THE SYSTEM DATA BASE AND DB2 INFORMATION USED WILL THEN DEPEND ON
* THE NAME BY WHICH THE PROGRAM IS INVOKED.
*
* THE STATEMENT " IIIDENT END" IS REQUIRED AS THE LAST SOURCE LINE.
*
*
IIBATCH  IIIDENT TYPE=IMS
IDBATCH  IIIDENT TYPE=DB2,SSID=DSN,PLAN=II,                            X
               NAME=DYLINQ.IISYSTEM
IVBATCH  IIIDENT TYPE=VSAM,SSID=DSN,PLAN=II,                           X
               NAME=IXXDB,LOCK=PASSWORD
         IIIDENT END
```

Figure 6-1 IXBIDENT Module with Necessary SMP/E control statements

You can use the following model JCL members provided in the II.PREP.CNTL data set to RECEIVE, APPLY, RESTORE, and ACCEPT the USERMODs for Identification modules. At most sites, there are ISPF-driven facilities that can also be used to perform these SMP/E processes.

JCL Member	Description
IQSMPE#A	RECEIVE a USERMOD into the Global Zone/Data sets.
IQSMPE#B	APPLY a USERMOD to the target libraries.
IQSMPE#C	ACCEPT a USERMOD to the distribution libraries.
IQSMPE#D	RESTORE (remove) a USERMOD from the target libraries.

Note: Once you ACCEPT an element, such as a USERMOD into the distribution libraries, there is no direct method for restoring the previous version of an element.

Use of Application and Terminal Names

To process an inquiry, VISION:Inquiry must know the inquiry name, the application name, and a terminal name to access the system database.

- The program uses the transaction identifier as the application and the originating terminal name (defined in Chapter 4, "The Definition Process") as the terminal (unless a user security exit changes this).
- The batch program gets this information by default or from /SET TRAN and /IAM LTERM commands (described later) in the input stream.

In either case, the inquiry programs use the application name and the terminal name to determine the following information from the system database (as described in Chapter 4, "The Definition Process"):

- Databases and fields that can be accessed, and the names used for them in the inquiry
- Vocabulary words that can be used in the inquiry
- Length and width of an output page
- BMS map used for output formatting
- Limitations, if any, on sorting
- Whether to use conversational or non-conversational (continuous) mode

Conversational Mode

In the conversational mode of operation that is designed for online processing, inquiry processing is controlled by the page length or maximum number of user database calls or VSAM reads permitted for the terminal.

When either a page end or the maximum number of calls or reads occurs, VISION:Inquiry stops processing the inquiry, displays any data that it has retrieved up to that point, and checkpoints the status and position information of the inquiry in the scratch pad storage area of the system database. At this point, you can either DEFER the inquiry for later processing or CONTINUE the inquiry for another set of pages of data or another maximum number of calls or reads.

- If the inquiry is deferred, VISION:Inquiry places the checkpoint information into the system database for the terminal and displays an identification number to be used when continuing the deferred inquiry.
- When an inquiry is continued, VISION:Inquiry uses the checkpoint information to reposition itself in the database or VSAM file and begins processing at the point of interruption, until a page end, maximum number of calls and reads, or end of inquiry occurs.

Conversational mode does not work well for batch inquiries, because there is no "terminal operator" to enter a CONTINUE command if one is needed. It is, however, ideal for online use; while you look over the response to your inquiry and decide what action to take, the CICS region is available to other users.

The same checkpoint logic (except DEFER command) applies to VISION:Journey during the extraction and writing data or report to the VISION:Journey download data set by VISION:Inquiry. The DEFER command is not supported for VISION:Journey.

Continuous Mode

Continuous mode is designed for batch processing. In continuous mode, an inquiry is processed until it is complete, or the maximum number of user database calls permitted is exceeded; then processing ends and inquiry can not be continued. This is not a good choice for use in CICS regions since resources are used continuously until the entire inquiry is complete.

If continuous mode is used in a CICS region for VISION:Journey, the data or report is written to the VISION:Journey download data set by VISION:Inquiry to the end of data or until the checkpoint call limit occurs. You can then download the partial data using VISION:Journey.

INQIO and IXXSTRT for Online Processing

The native VISION:Inquiry online processor operates in a CICS region and executes in the same manner as any other CICS application program. The online processor consists of two parts, the front-end (INQIO) and main (IXXSTRT) programs.

- The source of the front-end program (INQIO) is contained in II.TCVLSRC and can be modified according to your needs.
- IXCSORT is the other online program which is used by the native VISION:Inquiry online processor for sorting the output.

The online system uses the BMS paging facility for displaying output and the routing facility of the output command to send output to another terminal. The entries and JCL needed in the CICS tables and the CICS start-up JCL for running VISION:Inquiry are specified in *Advantage VISION:Inquiry for CICS Installation Guide*.

Additionally, the Text Editor facility runs in the online environment and uses the temporary storage and BMS facilities of CICS.

Intraccess only runs with the online version of VISION:Inquiry in the CICS region. See Intraccess Considerations on page 6-44 for more information about the programming and operational considerations for Intraccess.

VISION:Journey only runs with the online version of VISION:Inquiry in the CICS region; the JCL needed in the CICS start-up JCL for the download data set can be found in II.TCVLCNTL (CICSJCL).

Mapset Definition Considerations

Two sample mapsets are provided in II.TCVLSRC (INQMAP1, INQMAP2). The main difference between these two mapsets is the way they format the output, as discussed in Output Map Definition Specifications on page 6-14.

VISION:Inquiry uses the BMS paging facility to send the output of the inquiry to the terminal. If, in the terminal definition, no mapset name was specified in the BMS parameter, VISION:Inquiry uses a default mapset name specified in the member CVLCPARM in II.TCVLSRC as input and output. You can change the default mapset name, as described in the *Advantage VISION:Inquiry for CICS Installation Guide*. In the supplied CVLCPARM, the default mapset is INQMAP2.

The processing of VISION:Inquiry requires the default mapset to be present.

Input Map Definition Specifications

The input map, INQMAP2 (default), is used in the front-end program (INQIO), to format the screen for inquiry. The map must be consistent with the front-end program.

Output Map Definition Specifications

Two sample output mapsets are provided in II.TCVLSRC (INQMAP1 and INQMAP2). The names of the maps HEADING, FOOTING, FINAL, CHKPNT, DETLINE, and RTEFLIP are standard and must be present. The maps EDERR, START, LINES, and HEADA are used by front-end program only.

- The HEADING map is displayed at the top of every page except the last page. The P/N (page next) is optional. You can change the number of lines from the original query to show on each page of output. While the mapset INQMAP1 does not show any of the original query on the output, the mapset INQMAP2 shows the first four lines of the original query on each page. The front-end program adds an additional page at the end that contains the original query in its entirety.
- The map EDERR is used by the front-end program in case of error during the process of the query passed by the Text Editor.
- The maps START, LINES, and HEADA are used by the front-end program to format the screen for input and to build the last page of output which contains the original query.
- The FOOTING map is at the bottom of each page, except the last two pages. It must contain a five character output field to hold the BMS page number.
- The FINAL or CHKPNT map is displayed at the bottom of the next to last page of screen output.
- The HEADING map is required and must contain exactly one output field, 8 bytes long (if wanted, you can make this field “hidden” or dark).
- The FOOTING, FINAL, and CHKPNT maps are only required when the FOOTING=YES parameter is specified in the terminal definition.
- The RTEFLIP map is used when the user utilizes the OUTPUT command. (If you disable the OUTPUT keyword, the map is not required.)
- It is assumed that the maximum number of lines on any TRAILER=YES map is two lines. If you need to change it, modify the FOOTLEN parameter in the member CVLCPARM in II.TCVLSRC, as described in the *Advantage VISION:Inquiry for CICS Installation Guide*. If you have one mapset with trailer maps (FOOTING=YES) and another with no trailer maps (FOOTING=NO), you need to apply APAR058 to correct this.

Using the FOOTING=NO parameter does not make the bottom of the screen available for output unless you remove all trailer maps from the mapset.

If the CICS table entries are properly set and if appropriately suffixed BMS mapsets are created and defined, VISION:Inquiry can take advantage of the alternate screen size. However, to use the full width of a Model-5 terminal, you must specify a terminal definition width of 132.

You can use extended attributes (such as highlighting, color, or boxing), but use the EXTATT=MAPONLY parameter.

Additionally, VISION:Journey needs the BMS map INQMAP2 to be used by VISION:Inquiry for processing the inquiry. Very limited changes are allowed for this map (for example, the protected field values can be changed).

VISION:Journey also requires that the first three positions of the first line of the footing line at the bottom of the screen (if it exists) be blank.

Text Editor Map Definition Specifications

The mapsets used by Text Editor facility are in II.TCVLSRC (IQEMAPM and IQEMAPH).

- IQEMAPM is the main Text Editor screen.
- IQEMAPH is the Help screen which describes the function of the commands and PF keys used on the main screen.

You can change the mapset names, as described in the *Advantage VISION:Inquiry for CICS Installation Guide*. In the supplied member CVLCPARM in II.TCVLSRC, the mapset names are IQEMAPM (for main) and IQEMAPH (for Help).

You can also modify selected areas of the Text Editor maps:

- The contents and attributes of the fields of the Help screen, IQEMAPH.
- Add or delete the help pages. The HELPPG parameter of the source library member II.TCVLSRC (CVLCPARM) should reflect the number of help pages.
- The contents of the protected fields of the main screen, IQEMAPM.
- The order of the PF key functions on the last line of the main screen.
- The position of the fields on the first three lines of the main screen but not the order of the fields in the mapset member II.TCVLSRC (IQEMAPM). The only exception is the position of the COMMAND field on the first line which must not be changed.
- You can use extended attributes (such as highlighting or color) but use EXTATT=MAPONLY parameter.

You cannot modify:

- The size of the main and Help screen mapsets.
- The position of the COMMAND field in the main screen.
- The main body of IQEMAPM screen, such as lines 4 through line 23.
- The length of the unprotected fields in the main screen mapset.
- The functions of the PF keys.

Native VISION:Inquiry Programming Considerations

The VISION:Inquiry front-end program (INQIO) builds the IOAREA that will contain the default map name for output and the inquiry. It then passes it to the main program, IXXSTRT, for processing.

Building IOAREA with INQIO

The front-end program provided in II.TCVLSRC (INQIO) builds an IOAREA from a formatted screen with 20 lines of input data.

- The CICS GETMAIN command must be used to obtain the IOAREA.
- The IOAREA is freed by VISION:Inquiry before it returns to the front-end program. (The FREEMAIN will cause a CICS ABEND if you did not use a GETMAIN area.)
- The length of the IOAREA is 1600 bytes, but can be increased to a maximum of 4K.

Format of the IOAREA

The format of the IOAREA starts with a 7-byte area which is the name of the default mapset name, followed by a 4-byte inquiry prefix, followed by the inquiry. The first 2 bytes of the prefix contain the length of the inquiry, not including the prefix length (4 bytes). Bytes 3 and 4 are reserved and used internally by VISION:Inquiry.

The default mapset name is used to format the output of the inquiry if the BMS parameter is not specified in the terminal definition. However, the default map should always be present in the system even if the BMS parameter is specified because some error messages use the default map to send the output to the screen.

The format of the IOAREA is shown below.

DEFAULT MAPSET NAME	INQUIRY PREFIX	INQUIRY	
1	8	12	1600

The front-end program passes control to the main program using the CICS LINK command.

The front-end program name should be the same as the PSB name for the system database and user databases if DL/I is used. This is necessary because the system schedules the PSB using the default name. The COMMAREA parameter of the LINK command contains the address of the IOAREA.

Additionally for VISION:Journey users, if the extraction is done completely (such as no checkpoint happened), the VISION:Inquiry main routine sets the COMMAREA pointer passed by INQIO to zeroes. This pointer is checked in the INQIO program. If the pointer is zero, a normal CICS return is issued and the VISION:Journey transaction will be started automatically to download the extracted data.

The users can modify or rewrite the front-end program in any programming language as long as the considerations of this section are met, and pass the IOAREA with the above format to the main routine.

Figure 6-2 shows a sample COBOL front-end program and the BMS map used in the program.

The BMS map is as follows:

```
IQDEMOM  TITLE  'MAP FOR IQDEMO'
IQDEMOM  DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL,TERM=3270-2,       X
               CTRL=FREEKB,STORAGE=AUTO,TIOAPFX=YES
IQDEMOM  DFHMDI SIZE=(24,80),HEADER=YES,JUSTIFY=FIRST
*
         DFHMDF POS=(01,01),ATTRB=(UNPROT),LENGTH=10
*
         DFHMDF POS=(01,30),ATTRB=(ASKIP,NORM),LENGTH=16,              X
               INITIAL='EMPLOYEE LISTING'
*
         DFHMDF POS=(07,32),ATTRB=(ASKIP,NORM),LENGTH=10,              X
               INITIAL='PLANT NAME'
*
PLANT1   DFHMDF POS=(09,30),ATTRB=(IC,UNPROT),LENGTH=1,                X
               INITIAL=' '
         DFHMDF POS=(09,32),ATTRB=(ASKIP,NORM),LENGTH=12,              X
               INITIAL='DALLAS SALES'
*
PLANT2   DFHMDF POS=(11,30),ATTRB=(UNPROT),LENGTH=1,                   X
               INITIAL='_'
```

Figure 6-2 BMS Map Used in COBOL Front-End Program (Page 1 of 2)

```
         DFHMDF POS=(11,32),ATTRB=(ASKIP,NORM),LENGTH=23,             X
               INITIAL='REMOTE CONTROL PRODUCTS'
*
PLANT3   DFHMDF POS=(13,30),ATTRB=(UNPROT),LENGTH=1,                  X
               INITIAL=' '
         DFHMDF POS=(13,32),ATTRB=(ASKIP,NORM),LENGTH=23,             X
               INITIAL='CORPORATE HDQTRS DALLAS'
*
PLANT4   DFHMDF POS=(15,30),ATTRB=(UNPROT),LENGTH=1,                  X
               INITIAL=' '
         DFHMDF POS=(15,32),ATTRB=(ASKIP,NORM),LENGTH=15,             X
               INITIAL='DELUXE PRODUCTS'
*
PLANT5   DFHMDF POS=(17,30),ATTRB=(UNPROT),LENGTH=1,                  X
               INITIAL=' '
         DFHMDF POS=(17,32),ATTRB=(ASKIP,NORM),LENGTH=19,             X
               INITIAL='MECHANICAL PRODUCTS'
*
PLANT6   DFHMDF POS=(19,30),ATTRB=(UNPROT),LENGTH=1,                  X
               INITIAL=' '
         DFHMDF POS=(19,32),ATTRB=(ASKIP,NORM),LENGTH=10,             X
               INITIAL='BASIC TOYS'
*
PLANT7   DFHMDF POS=(21,30),ATTRB=(UNPROT),LENGTH=1,                  X
               INITIAL=' '
         DFHMDF POS=(21,32),ATTRB=(ASKIP,NORM),LENGTH=12,             X
               INITIAL='DISTRIBUTION'
*
         DFHMSD TYPE=FINAL
         END
```

Figure 6-2 BMS Map Used in COBOL Front-End Program (Page 2 of 2)

The front-end program displays a list of plant names for VSPLANT VSAM test file and you can select one of them. Then the program passes control to the main routine of VISION:Inquiry and executes the stored inquiry "INQY.TEST" with plant identification corresponding to plant name selected as a substitutable value.

```
*****************************************************************
*                                                               *
*            **  VISION:Inquiry DEMO PROGRAM  **               *
*                                                               *
*  IQCDEMO DEMONSTRATES THE PROCESS OF ACCESSING                *
*  VISION:Inquiry VIA A COBOL PROGRAM.                          *
*                                                               *
*****************************************************************
*                                                               *
*              **  IDENTIFICATION DIVISION  **                  *
*                                                               *
*****************************************************************

 IDENTIFICATION DIVISION.

 PROGRAM-ID.  IQCDEMO.
 AUTHOR.      Computer Associates.

 DATE-WRITTEN.  JULY, 1990.
 DATE-COMPILED.
*****************************************************************
*                                                               *
*                 **  ENVIRONMENT DIVISION  **                  *
*                                                               *
*****************************************************************

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-370.
```

Figure 6-3 Sample COBOL Front-End Program (Page 1 of 6)

```
 OBJECT-COMPUTER. IBM-370.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

*****************************************************************
*                                                               *
*                     **  DATA DIVISION  **                     *
*                                                               *
*****************************************************************

 DATA DIVISION.
 FILE SECTION.
     EJECT
*****************************************************************
*                                                               *
*                  **  WORKING-STORAGE SECTION  **              *
*                                                               *
*****************************************************************

 WORKING-STORAGE SECTION.

 01  WORK-AREAS.

     05  IQDEMOM                     PIC X(08) VALUE 'IQDEMOM'.
     05  IXXSTRT                     PIC X(08) VALUE 'IXXSTRT'.
     05  IDMO                        PIC X(04) VALUE 'IDMO'.
     05  IERR                        PIC X(04) VALUE 'IERR'.
     05  WS-BLANK                    PIC X(01) VALUE ' '.

     05  WS-PASS-AREA.
         10  WS-PASS-AREA-MAP        PIC X(07) VALUE 'INQMAP2'.
         10  WS-PASS-AREA-LEN        PIC S9(4) VALUE +23  COMP.
         10  FILLER                  PIC S9(4) VALUE +0   COMP.
         10  WS-PASS-AREA-IQ         PIC X(18) VALUE 'VSPLANT INQY.TEST '.
         10  WS-PASS-AREA-PLANT      PIC X(05) VALUE '00000'.

     05  WS-PASS-AREA-LENGTH         PIC S9(4) VALUE +34  COMP.
     EJECT
*****************************************************************
*                                                               *
*                     **  DEMO MAP  **                          *
*                                                               *
*****************************************************************

 COPY IQDEMOM.
     EJECT
 LINKAGE SECTION.

*****************************************************************
*                                                               *
*                   **  LINKAGE SECTION  **                     *
*                                                               *
*****************************************************************

 01  DFHCOMMAREA.
     05 COMM-AREA                    PIC S9(8)  COMP.

 01  PARMLIST.
     05 FILLER                       PIC S9(8)  COMP.
     05 PARM-POINTER                 PIC S9(8)  COMP.
 01  PASS-DATA                       PIC X(34).
     EJECT
```

Figure 6-3 Sample COBOL Front-End Program (Page 2 of 6)

```
*****************************************************************
*                                                               *
*                  **  PROCEDURE DIVISION  **                   *
*                                                               *
*****************************************************************

 PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA.

 000-MAIN.

     PERFORM 100-HANDLE THRU 199-EXIT.

     IF EIBCALEN > 0

        PERFORM 400-MAPIN THRU 499-EXIT

        PERFORM 500-GET-PLANT THRU 599-EXIT

        IF WS-PASS-AREA-PLANT NOT = '00000'
           PERFORM 600-GETMAIN       THRU 699-EXIT
           MOVE WS-PASS-AREA TO PASS-DATA
           PERFORM 700-LINK-INQ-PROG THRU 799-EXIT
           PERFORM 300-MAPOUT THRU 399-EXIT
        ELSE
           PERFORM 200-MAPOUT THRU 299-EXIT
     ELSE
        PERFORM 200-MAPOUT THRU 299-EXIT.

     EXEC CICS RETURN TRANSID(IDMO)
                      COMMAREA(COMM-AREA)
                      LENGTH(4)
     END-EXEC.
     EJECT
*****************************************************************
*                                                               *
*              HANDLE CONDITION                                 *
*                                                               *
*****************************************************************

 100-HANDLE.

     EXEC CICS HANDLE AID CLEAR(900-RETURN)
     END-EXEC.

     EXEC CICS IGNORE CONDITION EOC
     END-EXEC.

     EXEC CICS HANDLE CONDITION MAPFAIL(800-MAPFAIL)
                                ERROR(890-ERROR)
     END-EXEC.

 199-EXIT. EXIT.
     EJECT
```

Figure 6-3 Sample COBOL Front-End Program (Page 3 of 6)

```
*****************************************************************
*                                                               *
*              SEND MAP                                         *
*                                                               *
*****************************************************************

 200-MAPOUT.

     EXEC CICS SEND MAP(IQDEMOM)
                       MAPONLY
                       ERASE
     END-EXEC.

 299-EXIT. EXIT.


 300-MAPOUT.

     EXEC CICS SEND MAP(IQDEMOM)
                       MAPONLY
                       ERASE
                       PAGING
                       ACCUM
     END-EXEC.

 399-EXIT. EXIT.
     EJECT

*****************************************************************
*                                                               *
*              RECEIVE MAP                                      *
*                                                               *
*****************************************************************

 400-MAPIN.

     EXEC CICS RECEIVE MAP(IQDEMOM)
                       INTO(IQDEMOMI)
     END-EXEC.

 499-EXIT. EXIT.
     EJECT
```

Figure 6-3 Sample COBOL Front-End Program (Page 4 of 6)

```
*****************************************************************
*                                                               *
*              GET PLANT NUMBER                                 *
*                                                               *
*****************************************************************

500-GET-PLANT.
     IF PLANT1L NOT = 0
        MOVE '10100' TO WS-PASS-AREA-PLANT
     ELSE
        IF PLANT2L NOT = 0
           MOVE '20150' TO WS-PASS-AREA-PLANT
        ELSE
           IF PLANT3L NOT = 0
              MOVE '30200' TO WS-PASS-AREA-PLANT
           ELSE
              IF PLANT4L NOT = 0
                 MOVE '40300' TO WS-PASS-AREA-PLANT
              ELSE
                 IF PLANT5L NOT = 0
                    MOVE '50300' TO WS-PASS-AREA-PLANT
                 ELSE
                    IF PLANT6L NOT = 0
                       MOVE '60200' TO WS-PASS-AREA-PLANT
                    ELSE
                       IF PLANT7L NOT = 0
                          MOVE '70500' TO WS-PASS-AREA-PLANT
                       ELSE
                          MOVE '00000' TO WS-PASS-AREA-PLANT.

 599-EXIT. EXIT.
     EJECT

*****************************************************************
*                                                               *
*              GETMAIN                                          *
*                                                               *
*****************************************************************


 600-GETMAIN.

     EXEC CICS GETMAIN LENGTH(WS-PASS-AREA-LENGTH)
                       INITIMG(WS-BLANK)
                       SET(PARM-POINTER)
     END-EXEC.

 699-EXIT. EXIT.
     EJECT
*****************************************************************
*                                                               *
*              LINK TO INQ PROGRAM                              *
*                                                               *
*****************************************************************

 700-LINK-INQ-PROG.

     EXEC CICS LINK PROGRAM(IXXSTRT)
                    COMMAREA(PARM-POINTER)
                    LENGTH(4)
     END-EXEC.
 799-EXIT. EXIT.
     EJECT
```

Figure 6-3 Sample COBOL Front-End Program (Page 5 of 6)

```
*****************************************************************
*                                                               *
*              MAPFAIL                                          *
*                                                               *
*****************************************************************

 800-MAPFAIL.

     PERFORM 200-MAPOUT THRU 299-EXIT.

     EXEC CICS RETURN TRANSID(IDMO)
                      COMMAREA(COMM-AREA)
                      LENGTH(4)
     END-EXEC.

*****************************************************************
*                                                               *
*              ERROR                                            *
*                                                               *
*****************************************************************

 890-ERROR.

     EXEC CICS ABEND ABCODE(IERR)
     END-EXEC.

     EJECT
*****************************************************************
*                                                               *
*              RETURN TO CICS, NO RETURN                        *
*                                                               *
*****************************************************************
*
 900-RETURN.

     EXEC CICS RETURN
     END-EXEC.


     GOBACK.
```

Figure 6-3 Sample COBOL Front-End Program (Page 6 of 6)

VISION:Inquiry Messages and Dump Considerations

VISION:Inquiry issues the CICS ABEND command, PLIDUMP, and error messages for error conditions that occur during inquiry processing.

VISION:Inquiry ABEND Codes

The error conditions, the modules that caused the error, and the ABEND codes are as follows:

ERROR CONDITION	MODULE NAME	ABEND CODE
INVMPSZ	IXCINIT,	IVSZ
INVREQ	IXCOMSG	IVRQ
TSIDERR	IXCINIT,	TSER
	IXCOMSG	DBER
	IXCINIT, IXCOMSG IXCDBER	This ABEND occurs when there is a system database error. A message is sent to the terminal showing the UIB or PCB status code.

Messages and Dump

VISION:Inquiry issues some informational messages and also a PLIDUMP for some errors during processing. The transient data set CESE is used for the messages and dump under CICS.

Terminal Input Format

You can start VISION:Inquiry and pass the inquiry string by starting a task that uses the front-end program to send a map to format the screen. This allows you to enter your transaction ID and inquiry in the appropriate fields.

If your output map is designed so that it does not overlap your input map, you can use the same input map for subsequent inquiries.

Terminal Output Format

Output from VISION:Inquiry appears in one of two formats designed to take advantage of output device capabilities. If the output device line width is sufficient to accommodate all data requests, the output is presented in column or vertical format with headings. If the device line width is insufficient to accommodate the data requested, the output is presented in the format of a field name followed by the field contents on the same line, with each field beginning on a new line called row or horizontal format.

The examples below show both formats as they appear for the same inquiry on devices with different line widths:

Vertical Format

```
EMPLOYEE              LAST.NAME             SKILL

01401                 SMITH                 CARPENTER
                                            STEAM FITTER
                                            SALESMAN
```

Horizontal Format

```
EMPLOYEE                = 01401
       LAST NAME        = SMITH
       SKILL            = CARPENTER
       SKILL            = STEAM FITTER
       SKILL            = SALESMAN
```

In vertical format, the order of headings from left to right is the order of the object fields in the inquiry statement for databases. The order of retrieval is determined by the hierarchical structure of the database and is the order of output in horizontal output format.

The User-Defined Output (UDO) feature of VISION:Inquiry allows users to format the output. They can determine column headings and placement of data and literals on the screen.

Because VISION:Inquiry uses the CICS paging facility to send output, this must be provided for on your system. To page forward or backward, you can use the paging commands and PF keys according to the definitions specified on your System Initialization Table (SIT).

The IIBATCH Program for Batch

The inquiry processor, IIBATCH, operates in ordinary batch or IMS (DL/I) batch.

IIBATCH Input

Input to IIBATCH is from a sequential data set, PDS member, or system input device of fixed-length, 80-byte records assigned to ddname SYSIN.

The format of the input inquiry is identical except that the logical terminal name precedes the transaction code. For example,

```
BATCH IQIO DISPLAY PLANT EMP.NO EMP.NAME;;
```

where BATCH is the logical terminal name and IQIO the transaction code.

Transaction names, logical terminal names, and the mode of operation can be set for the entire job by using the appropriate set of the following commands.

```
/SET TRAN transaction-name
/IAM lterm - name

/SET ONLINE (starts conversational mode)

/SET BATCH (stops conversational mode)
```

Specifying /SET TRAN and /IAM commands prior to submitting inquiries eliminates the need to specify the transaction code and terminal name in the individual inquiry. However, even if these are set as stated, you can still specify a different transaction code or terminal name in each inquiry. For example, multiple inquiries may be entered and each may specify a different or the same transaction code.

- Terminate inquiries in batch mode by two semicolons (;;).
- Terminate single statements of a multi-statement inquiry with one semicolon (;).

The entire 80-byte logical record of each SYSIN file is examined for inquiry input. If the input file contains sequence numbers in any of the columns 73 through 80, unpredictable results in the processing of those inquiries may result. To resolve this problem, enter the command /SEQ, beginning in column 1, in the input stream at the point where sequence numbers occur. This causes VISION:Inquiry to examine only columns 1 through 72 of the input records.

The command /NOSEQ can be entered in the input stream at any point to reset the examination of the records to the entire 80 bytes. /SEQ and /NOSEQ can be used intermittently within the input stream as often as required to set or reset the examination of columns 73 through 80.

IIBATCH Output

Output is to a sequential data set (or system output device) of fixed-length, 133-byte records with ANSI control characters assigned to ddname SYSOUT.

IIBATCH Execution

You can execute IIBATCH in a DL/I batch region (using the IMS procedure DLIBATCH). The PSB used is the same as for online operation.

You can also execute IIBATCH in an ordinary (non-DL/I) region if you are not using an IMS (DL/I) system database or IMS (DL/I) user databases.

IIBATCH Execution JCL to Access IMS (DL/I) System and User Databases

The VISION:Inquiry batch program, IIBATCH, follows the conventions of DLIBATCH execution. The program name is IIBATCH and an application PSB is required. A DD statement is required for the system database and any of your databases to be queried.

II.TCVLCNTL (IQBATD and IQBATD2)

The control library member II.TCVLCNTL (IQBATD), as illustrated in Figure 6-4 on page 6-27, can be used as a guideline for VISION:Inquiry batch mode operation to access IMS (DL/I) system and user databases.

In the example JCL, the test databases, distributed with the VISION:Inquiry system, are accessed. Also included in the JCL stream is the VISION:Inquiry system database. Input to the batch inquiry processor (IIBATCH) is performed by SYSIN. Output from the inquiries is printed using SYSOUT.

```
//*
//* THIS IS A SAMPLE JCL FOR EXECUTING THE BATCH VERSION OF THE
//* PRODUCT TO ACCESS IMS(DL/I)  SYSTEM AND/OR USER DATABASES.
//* IT USES THE STANDARD DLIBATCH PROCEDURE.
//*
//* IT IS ALSO USED AS JCL TO RESTORE INQUIRIES AND FUNCTIONS UNLOADED
//* BY THE IXUSQRY OR IXUIQRY UTILITY INTO THE IMS(DL/I)  SYSTEM
//* DATA BASE. THE SYSIN DD STATEMENT SHOULD REFERENCE THE UNLOADED
//* DATASET FOR THIS CASE.
//*
//* BEFORE RUNNING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DD AND DSNAMES AS NEEDED.
//*   - ADD THE DD STATEMENTS FOR THE USER DATABASES.
//*   - VERIFY OR CHANGE THE INPUT CONTROL STATEMENTS AND
//*     INQUIRY AS NEEDED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETED.
```

Figure 6-4 JCL to Access IMS (DL/I) System and User Databases (IQBATD) (Page 1 of 2)

```
//*
//IIBATCH  EXEC DLIBATCH,MBR=IIBATCH,PSB=INQIO,REGION=2500K
//STEPLIB  DD
//         DD   DSN=II.TCVLPGM,DISP=SHR
//IMS      DD   DSN=II.PSBLIB,DISP=SHR
//         DD   DSN=II.DBDLIB,DISP=SHR
//IEFRDER  DD   DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//DFSVSAMP DD   *  SEE YOUR DBA FOR RIGHT CHOICE OF CONTROL STATEMENTS
16384,6
1024,6
//*           SORT (WORK) FILES (IF USING SORT COMMAND)
//SORTLIB  DD   DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK02 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK03 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//*           EXTRACT FILE (IF USING EXTRACT COMMAND)
//IXXEXTRA DD   DSN=&&EXTRACT,DISP=(,PASS),UNIT=VIO,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//*           YOUR SYSTEM DATABASE
//*           REMOVE THE FOLLOWING DD STATEMENT IF DB2 SYSTEM DATABASE
//IXXDB    DD   DSN=II.IXXDB,DISP=OLD
//*           YOUR DATABASES GO HERE (UNLESS SHARING VIA BMP)
//*
//IIDBDDM  DD DSN=II.PLANT,DISP=SHR       TEST PLANT FILE
//IIDBDDMO DD DSN=II.PLANTOV,DISP=SHR
//IIDBDDS  DD DSN=II.SKILL,DISP=SHR       TEST SKILL FILE
//IIDBDDSO DD DSN=II.SKILLOV,DISP=SHR
//*           YOUR VSAM FILES GO HERE (IF ANY)
//PLIDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD   *  CONTROL STATEMENTS GO HERE
/SEQ
/IAM LTERM BATCH
/SET TRAN IQIO
 D PLANT PLANT.ID;;
```

Figure 6-4 JCL to Access IMS (DL/I) System and User Databases (IQBATD) (Page 2 of 2)

The control library member II.TCVLCNTL(IQBATD2), as illustrated in Figure 6-5, can be used as a guideline for a VISION:Inquiry IMS (DL/I) batch mode operation to access DB2 system or user databases. In this mode of operation, VISION:Inquiry uses IMS attach to make a connection to DB2.

```
//*
//* THIS IS A SAMPLE JCL FOR EXECUTING THE BATCH VERSION OF THE
//* PRODUCT TO ACCESS DB2 SYSTEM OR USER TABLES UNDER IMS BATCH.
//* IT USES THE STANDARD DLIBATCH PROCEDURE.
//*
//* IT IS ALSO USED AS JCL TO RESTORE INQUIRIES AND FUNCTIONS UNLOADED
//* BY THE IXUSQRY OR IXUIQRY UTILITY INTO THE DB2 SYSTEM DATA BASE.
//* THE SYSIN DD STATEMENT SHOULD REFERENCE THE UNLOADED DATASET FOR
//* THIS CASE.
//*
//* IT IS ALSO USED AS JCL TO RESTORE INQUIRIES AND FUNCTIONS UNLOADED
//* BY THE IXUSQRY OR IXUIQRY UTILITY INTO THE DB2 SYSTEM DATA BASE.
//* THE SYSIN DD STATEMENT SHOULD REFERENCE THE UNLOADED DATASET FOR
//* THIS CASE.
//*
//* BEFORE RUNNING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DD AND DSNAMES AS NEEDED.
//*   - ADD THE DD STATEMENTS FOR THE USER DATA BASES.
//*   - VERIFY OR CHANGE THE INPUT CONTROL STATEMENTS AND
//*     INQUIRY AS NEEDED.
```

Figure 6-5 JCL to Access DB2 System or User Databases in IMS Batch (IQBATD2) (Page 1 of 2)

```
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE.
//*
//IIBATCH  EXEC DLIBATCH,MBR=DSNMTV01,PSB=INQIO,REGION=2500K
//STEPLIB  DD
//         DD   DSN=II.TCVLPGM,DISP=SHR
//         DD   DSN=DSN510.SDSNLOAD,DISP=SHR
//IMS      DD   DSN=II.PSBLIB,DISP=SHR
//         DD   DSN=II.DBDLIB,DISP=SHR
//IEFRDER  DD   DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//DFSVSAMP DD   *  SEE YOUR DBA FOR RIGHT CHOICE OF CONTROL STATEMENTS
16384,6
1024,6
//*           DB2 SUBSYSTEM DEFINITION
//*   VERIFY AND CHANGE THE DDITV02 INPUT INFORMATION IF NECESSARY OR
//*   CONSULT WITH YOUR DB2 ADMINISTRATOR FOR ALTERNATIVE TECHNIQUES
//*
//DDITV02  DD  *
DSN,,DSNMIN10,,,,,II,IIBATCH
//*
//DDOTV02  DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(1,1)),
//             DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//*
//*           SORT (WORK) FILES (IF USING SORT COMMAND)
//SORTLIB  DD   DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK02 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK03 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//*           EXTRACT FILE (IF USING EXTRACT COMMAND)
//IXXEXTRA DD   DSN=&&EXTRACT,DISP=(,PASS),UNIT=VIO,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//*           YOUR SYSTEM DATA BASE
//*           REMOVE THE FOLLOWIG DD STATEMENT IF DB2 SYSTEM DATA BASE
//IXXDB    DD   DSN=II.IXXDB,DISP=OLD
//*           YOUR DATA BASES GO HERE (UNLESS SHARING VIA BMP)
//*
//IIDBDDM  DD DSN=II.PLANT,DISP=SHR       TEST PLANT FILE
//IIDBDDMO DD DSN=II.PLANTOV,DISP=SHR
//IIDBDDS  DD DSN=II.SKILL,DISP=SHR       TEST SKILL FILE
//IIDBDDSO DD DSN=II.SKILLOV,DISP=SHR
//*           YOUR VSAM FILES GO HERE (IF ANY)
//PLIDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD   *  CONTROL STATEMENTS GO HERE
/SEQ
/IAM LTERM BATCH
/SET TRAN IQIO
D PLANT PLANT.ID;;
```

Figure 6-5 JCL to Access DB2 System or User Databases in IMS Batch (IQBATD2) (Page 2 of 2)

The DDITV02 input file in Figure 6-5 is used to specify the DB2 subsystem ID (DSN), the plan name (II), and the load module name (IIBATCH). You can also use other techniques to specify this information.

IIBATCH Execution JCL to Access Non-IMS System and User Databases

The VISION:Inquiry batch program, IIBATCH, can be executed as an ordinary batch job to access the DB2 or VSAM system database, user DB2 tables, and user VSAM data sets.

II.TCVLCNTL (IQBATV)

The control library member II.TCVLCNTL (IQBATV), as illustrated in Figure 6-6, is batch JCL which can be used as a guideline for VISION:Inquiry batch mode operation.

In the example JCL, the VSAM test files, distributed with the VISION:Inquiry system, are accessed. Input to the batch inquiry processor (IIBATCH) is performed by SYSIN. Output from the inquiries is printed using SYSOUT.

```
//*
//* THIS IS A SAMPLE JCL FOR EXECUTING THE BATCH VERSION OF THE
//* PRODUCT TO ACCESS NON IMS(DL/I) SYSTEM DATABASE & USER DATA BASES.
//*
//* IT IS ALSO USED AS JCL TO RESTORE INQUIRIES AND FUNCTIONS UNLOADED
//* BY THE IXUSQRY OR IXUIQRY UTILITY.  THE SYSIN DD STATEMENT SHOULD
//* REFERENCE THE UNLOADED DATASET FOR THIS CASE.
//*
//* BEFORE RUNNING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DD AND DSNAMES AS NEEDED.
//*   - ADD THE DD STATEMENTS FOR THE USER VSAM FILES (IF ANY).
//*   - VERIFY OR CHANGE THE INPUT CONTROL STATEMENTS AND
//*     INQUIRY AS NEEDED.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETED.
//*
//IIBATCH  EXEC PGM=IIBATCH,REGION=2500K
//STEPLIB  DD   DSN=II.TCVLPGM,DISP=SHR
//         DD   DSN=DSN510.SDSNLOAD,DISP=SHR    FOR DB2 TABLES ACCESS
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//*           SORT (WORK) FILES (IF USING SORT COMMAND)
//SORTLIB  DD   DSN=SYS1.SORTLIB,DISP=SHR
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK02 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//SORTWK03 DD   UNIT=SYSDA,SPACE=(CYL,(3,1),,CONTIG)
//*           EXTRACT FILE (IF USING EXTRACT COMMAND)
//IXXEXTRA DD   DSN=&&EXTRACT,DISP=(,PASS),UNIT=VIO,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096)
//*           YOUR VSAM SYSTEM DATA BASE
//*           REMOVE THE FOLLOWIG DD STATEMENT IF DB2 SYSTEM DATA BASE
//IXXDB  DD DSN=II.IXXDB,DISP=OLD
//*           YOUR VSAM FILES (IF ANY) MUST GO HERE
//VSPLDS   DD   DSN=VS.VSPLDS,DISP=SHR   TEST VSPLANT FILE
//VSSKDS   DD   DSN=VS.VSSKDS,DISP=SHR   TEST VSSKILL FILE
//PLIDUMP  DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//SYSIN    DD   *  CONTROL STATEMENTS GO HERE
/SEQ
/IAM LTERM BATCH
/SET TRAN IQIO
 D VSPLANT VSPLANT.ID;;
```

Figure 6-6 JCL to Access Non-IMS (DL/I) System and User Databases (IQBATV)

Batch Sorting with the SORT Command

The SORT command provides you with the ability to reorganize the data you select for display.

When processing an inquiry containing a SORT command, VISION:Inquiry passes selected data (based on the DISPLAY fields and conditional selection) to the installation sort routine. Once sorted, this data is returned to VISION:Inquiry to be formatted and sent to the appropriate terminal or output device.

Allocate Sort Files

In order to accommodate user sorting requirements, the appropriate sort files must be allocated regardless of the mode of operation. Those files that are necessary include:

SORTWK01	SORTWK03	SORTLIB
SORTWK02	SORTWKnn	SYSOUT

SORTWK01 through SORTWKnn must be allocated with sufficient space to sort the maximum selected data if you do not sort in memory. The installation sort routine, found in the specified SORTLIB file, is used by VISION:Inquiry. The sort files are allocated in the JCL invoking the batch job.

Batch Extraction with the EXTRACT Command

You can use the EXTRACT command to extract data from an IMS (DL/I) database, DB2 table, or VSAM data set and write it to an OS sequential data set. The EXTRACT command syntax is exactly the same as the DISPLAY command.

- When EXTRACT is specified, an OS data set with a ddname of IXXEXTRA is opened and the specified database fields are formatted into a variable length record and written to the opened data set. A complete explanation of the record format appears below.
- The EXTRACT command provides a means of obtaining data from an IMS (DL/I) database, DB2 table, or VSAM data set in a format that can be sorted and read by report generators or other general purpose programs that use OS data sets. For example, VISION:Eighty™ or VISION:Results™ can be used to read the data set.

- The extract data set, IXXEXTRA, is only opened once per invocation of VISION:Inquiry. This means that the results of multiple EXTRACT commands are written back-to-back on the data set.
- Sorting is not permitted with EXTRACT. LIMIT may be used with EXTRACT.
- If a DD statement for the extract data set is not present in the job, a diagnostic message results.

Extract Data Set Format

When the EXTRACT command is used, the data set IXXEXTRA is opened and the extracted data is written to it as a variable length record. Every record for a single extract is the same length. The data set, however, is not reopened for second and subsequent inquiries for the same invocation of VISION:Inquiry. This means that multiple record formats can appear on the same extract data set.

The record format (RECFM) of the extract data set is variable blocked (VB). You must supply the block size (BLKSIZE) and logical record length (LRECL) in the data control block (DCB) for the extract data set.

How the Format is Determined

The format of the record is determined by examining the inquiry. Each extracted item occupies a fixed location in the record. The fields appear left to right just as they appear in the inquiry. To facilitate their use in subsequent report generation or sorting, higher level fields in the database are repeated for each occurrence of a lower level field.

Fields Extracted in Internal Form

All fields are extracted in their internal form. No conversion is made. When temporary fields that are the result of arithmetic operations are extracted, they are in packed format with a length of eight bytes.

- If the temporary field is used in an arithmetic calculation that involves multiplication or division, it has four decimal places.
- If only addition or subtraction is involved, the temporary field has two decimal places.
- If the temporary field assignment involves no arithmetic operations, the temporary field is extracted in the internal form of the assigned field.

Extract Data Set Results

The example that follows shows the EXTRACT command and its results.

Inquiry:

```
EXTRACT PLANT PLANT.ID PLANT.NAME EMP.NO EMP.SEX EMP.NAME;;
```

Results:

The column headings across the top are for position illustration and are not part of the extract data set.

```
000000000111111111122222222223333333333444444444455555555
123456789012345678901234567890123456789012345678901234567


10100ATLANTA DISTRIBUTION     10050FMARY ANN THOMAS
10100ATLANTA DISTRIBUTION     10100MWILLIAM AMES
20150CHICAGO SALES            20327FWILMA FORD
20150CHICAGO SALES            20850MRUSSELL SALTER
20150CHICAGO SALES            20900MPETER ZATKIN
30200CORPORATE HEADQUARTERS   30175MWILLIAM CRANE
30200CORPORATE HEADQUARTERS   30425MMITCHELL HOOPS
30200CORPORATE HEADQUARTERS   30450MFREDERICH GRAY
30200CORPORATE HEADQUARTERS   30500FPATRICIA LOW
30200CORPORATE HEADQUARTERS   30507FJANE LOWELL
30200CORPORATE HEADQUARTERS   30625FJUDITH PAULET
40300DALLAS RESEARCH CENTER   40250FJOAN EVANS
40300DALLAS RESEARCH CENTER   40450MDONALD KING
50350EASTERN ADMINISTRATION   50125FMARY JEAN BATES
50350EASTERN ADMINISTRATION   50620MWILLIAM OAKS
50350EASTERN ADMINISTRATION   50725FVICKY WARD
60375SEATTLE DISTRIBUTION CENT60327FWILMA MORINO
60375SEATTLE DISTRIBUTION CENT60850MRUSSELL SMITH
60375SEATTLE DISTRIBUTION CENT60950MPETER ZORK
```

Figure 6-7

Extracting Data From Multiple Databases

You can also use the INTER (FIND) command with the EXTRACT command to extract data from multiple databases or files.

Inquiry 1:

```
EXTRACT PLANT EMP.NO EMP.NAME ED.YEAR ED.DEGREE ED.SCHOOL
IF EMP.SEX = 'F';;
```

Inquiry 2:

```
INTER SKILL %X = EMP.NO %SKILL.NAME = SKILL.NAME;
EXTRACT PLANT %SKILL.NAME EMP.NO EMP.NAME IF EMP.NO = %X;;
```

Since two inquiries are input, the extracted data for both inquiries appears back-to-back in the extract data set.

Online Extraction with the EXTRACT Command

You can use the EXTRACT command to extract database or VSAM fields from a DL/I database, DB2 table, or VSAM file and write them to the extra-partition transient data set. The EXTRACT command syntax is exactly the same as the DISPLAY command.

- When EXTRACT is specified, the specified database or VSAM file fields are formatted into a variable length record and written to the extra-partition data set with the supplied destination ID of DYL1. However, the EXTRACT user exit (discussed in Chapter 7, "User Exits") can be used to modify the destination ID (DYL1) or to assign different destination IDs to different users.
- The EXTRACT command provides a means of obtaining data from a DL/I database, DB2 table, or VSAM file in a format that can be sorted and read by report generators or other general purpose programs that use MVS data sets. For example, VISION:Eighty™ or VISION:Results™ can be used to read the data set. The LIMIT command is used with EXTRACT.

Extract Data Set Format

When the EXTRACT command is used, the extracted data is written to the extra-partition data set, with the supplied name of DYL1, as a variable length record. Every record for a single extract is the same length. Depending on the OPEN parameter specification in the Destination Control Table (DCT), the data set will be opened either during CICS system start-up or by the master terminal command, before using the EXTRACT command. However, the data set is not reopened for second and subsequent inquiries.

This means that multiple record formats can appear on the same extract data set unless you use the EXTRACT user exit routine (discussed in Chapter 7, "User Exits") and assign multiple extract data sets to users.

The record format (RECFM) of the extract data set is variable blocked (VB). You must supply the block size (BLKSIZE) and logical record length (LRECL) in the DCT for the extract data set.

How the Format is Determined

The format of the record is determined by examining the inquiry. Each extracted item occupies a fixed location in the record. The fields appear left to right, just as they appear in the inquiry. For databases to facilitate their use in subsequent report generation or sorting, higher level fields in the database are repeated for each occurrence of a lower level field.

Fields Extracted in Internal Form

All fields are extracted in their internal form. No conversion is made. When temporary fields that are the result of arithmetic operations are extracted, they are in packed format with a length of eight bytes.

- If the temporary field is used in an arithmetic calculation that involves multiplication or division, it has four decimal places.
- If only addition or subtraction is involved, the temporary field has two decimal places.
- If the temporary field assignment involves no arithmetic operations, the temporary field is extracted in the internal form of the assigned field.

Extract Data Set Results

The example that follows shows the results of the EXTRACT command:

Inquiry:

```
EXTRACT PLANT PLANT.ID PLANT.NAME EMP.NO EMP.SEX EMP.NAME
```

The column headings across the top are for position illustration and are not part of the data set.

```
000000000111111111122222222223333333333444444444455555555
123456789012345678901234567890123456789012345678901234567

10100ATLANTA DISTRIBUTION     10100MARRY ANN THOMAS
10100ATLANTA DISTRIBUTION     10100MWILLIAM AMES
20150CHICAGO SALES            20327FWILMA FORD
20150CHICAGO SALES            20850MRUSSELL SALTER
20150CHICAGO SALES            20900MPETER ZATKIN
30200CORPORATE HEADQUARTERS   30175MWILLIAM CRANE
30200CORPORATE HEADQUARTERS   30425MMITCHELL HOOPS
30200CORPORATE HEADQUARTERS   30450MFREDERICH GRAY
30200CORPORATE HEADQUARTERS   30500FPATRICIA LOW
30200CORPORATE HEADQUARTERS   30507FJANE LOWELL
30200CORPORATE HEADQUARTERS   30625FJUDITH PAULET
40300DALLAS RESEARCH CENTER  40250FJOAN EVANS
40300DALLAS RESEARCH CENTER  40450MDONALD KING
50350EASTERN ADMINISTRATION   50125FMARY JEAN BATES
50350EASTERN ADMINISTRATION   50620MWILLIAM OAKS
50350EASTERN ADMINISTRATION   50725FVICKY WARD
60375SEATTLE DISTRIBUTION CENT60327FWILMA MORINO
60375SEATTLE DISTRIBUTION CENT60850MRUSSELL SMITH
60375SEATTLE DISTRIBUTION CENT60950MPETER ZORK
```

Figure 6-8

Extracting Data From Multiple Databases

You can also use the INTER (FIND) command with EXTRACT command to extract data from multiple databases or files.

Inquiry:

```
INTER SKILL %X = EMP.NO %SKILL.NAME = SKILL.NAME;
EXTRACT PLANT %SKILL.NAME EMP.NO EMP.NAME IF EMP.NO=%X;
```

Since two inquiries are input, the extracted data for both inquiries appears back-to-back in the extract data set.

Online Sorting with the SORT Command

The SORT command provides you with the ability to reorder the output data.

When the SORT command is encountered, PAGE and TIME limit parameters of the terminal definition statement do not interrupt SORT processing and the inquiry is processed to completion or until it reaches to the maximum value allowed either by SLIMIT or SORTSIZ parameter.

SORTSIZ parameter

The SORTSIZ parameter, in the member CVLCPARM in II.TCVLSRC, specifies the maximum number of records to be sorted online. The parameter is mainly used for performance purposes. If the number of records to be sorted exceeds the maximum, the sort program issues an error message and terminates processing. Up to eight digits are allowed. However, the actual maximum number of records that can be sorted depends on the following factors, and the number of records decreases as:

- The length of the sort fields increases.
- The output record length increases.
- The distribution of the sort fields in the user data set/database gets more random.

Sorting Large Number of Records

For example, the CICS sort routine can sort 268,100 records when the lengths of the output record and the sort fields are 81 and 25 bytes, respectively. However, Computer Associates highly recommends using the batch version of the product for the large number of records to be sorted.

When processing an online inquiry containing a SORT command, VISION:Inquiry passes selected records (based on the DISPLAY fields and conditional selection) to an internal sort routine; the standard SORT is not available under CICS. Once sorted, the data is returned to VISION:Inquiry to be formatted and sent to the appropriate terminal or output device.

Sorting Techniques

You can select one of the three sort techniques in the SORTIOM parameter, in the member CVLCPARM in II.TCVLSRC, for online sorting. Based on the selected technique, the records will be sorted using:

- Main storage
- Auxiliary temporary storage
- VSAM work data set

However, the sort fields will always be kept and sorted in the main storage no matter which technique is used. Comparing the three sort techniques, the main differences are in the response time and the main storage requirements.

- Using main storage for online sort requires more main storage (more stress on CICS) but has the fastest response time.
- Using auxiliary temporary storage is slower and requires a temporary storage data set for keeping the records.
- Using VSAM work data set gives you the slowest response time and requires a VSAM work data set to keep the records.

Sorting Considerations

When selecting the sorting technique, consider the following factors in your system:

- The number of records to be sorted.
- The availability of the main storage for the CICS region.
- The expected response time from sort.

For the first two considerations, the main storage technique requires much more main storage and puts more stress on the CICS system as the number of sorted records increases. However, regarding the third consideration, the main storage technique is much faster than the other two techniques.

When using the VSAM work data set technique for online sort, the sort routine will delete the records written to the work data set by the previous inquiry prior to starting the new sort which will slow down the response time considerably. To get around this problem, a user sort exit has been introduced which will be started at the end of each inquiry sort processing.

Introducing the User Sort Exit

The exit routine, IXCUSRT, is written to start a new transaction to delete the records from the VSAM work data set at the end of each inquiry sort processing. Two sample routines for the sort exit are provided in the II.TCVLSRC, members IXCUSRTS and IXCUSDL. Chapter 7, "User Exits" and Appendix B, "Sample User Exits" contain more detailed information on the sort exit.

Automatic Query Facility (AQF) Considerations

The following subsections describe some programming and operating techniques and considerations regarding generation and execution of inquiries using the AQF facility.

AQF Programming Considerations

The interface between the front-end program (INQIO) and AQF is as follows:

- AQF will (EXEC CICS) START the native VISION:Inquiry transaction passing it data.
- VISION:Inquiry will realize that it was started by AQF by the presence of "DYLBRDGE" in the first 8 bytes of the communication area. VISION:Inquiry will then (EXEC CICS) RETRIEVE the data sent to it.
- The data passed from AQF to native VISION:Inquiry has the following format:
  - Bytes 1 to 8 contain the original AQF transaction identifier (8 bytes are used, not 4, so that IMS can share the same format).
  - Bytes 9 and 10 contain the length of the first line actually filled in by AQF (AQF does not split words between lines).
  - Bytes 11 to 88 contain the first line of the query produced by AQF. (Excess characters are binary zeroes=low-values to allow the user to use the insert character key).
  - The last two fields are repeated (in pairs) up to 19 more times (a total of 20 lines can be generated by AQF).

- The front-end program (INQIO) shows the user the generated query and gives a chance to modify it (such as add parentheses around the IF conditions) before executing it. Notes are in the program source (II.TCVLSRC (INQIO)) if you want to execute the query immediately without giving the user a chance to modify the generated query. You could even have two programs and two (or more) transaction identifiers so that one transaction identifier would give you a chance to change the generated query and another would not.
- In order for native VISION:Inquiry to switch back to the AQF transaction, it must (EXEC CICS) START the AQF transaction. (Note that since you could have more than one AQF transaction, you should use the AQF transaction identifier originally passed to you when possible). You must make sure the first 8 bytes of the communication area does not contain `DYLBRDGE.' AQF will give the end user a choice of where to continue.
- There is also code in INQIO to switch from one native VISION:Inquiry transaction identifier to another VISION:Inquiry transaction (or any other CICS transaction for that matter). Currently, the program does not verify that the transactions exist and that the user is allowed to run that transaction (such as typing errors in the TRANCODE field can cause CICS ABENDs to occur).

AQF and CICS Temporary Storage Queues

AQF uses CICS temporary storage queues as work files. For any terminal, the queue name used is DYLTxxxx where xxxx is the CICS terminal ID. If the temporary storage queue disappears for any reason (such as PURGEd by someone using the CEBR transaction), the next time the user enters AQF, he or she is always forced into the introductory screen.

AQF Temporary Storage Queue

There are various records kept in the storage queue. All record sizes are minimized. If a storage queue record is no longer needed, it is rewritten with length 0. The various record types are:

Q-REC-1 Control record (always item #1), length 112.

Q-REC-2 One for each database selected (created when the select fields screen is entered).

Minimum length:	20 bytes
Maximum length:	900 bytes
Any SORT fields present:	minimum length: 504 bytes*
Any List order fields present:	minimum length: 100 bytes*
Any Control break fields present:	minimum length: 64 bytes*
Any Matching KEY fields present:	minimum length: 24 bytes

* The two databases act independently, for example, if all SORT fields come from one database, only the Q-REC-2 for that database will be forced up to 504-bytes.

Q-REC-3 Contains information for each field of the database. Each record contains information up to 27 fields.
Minimum length: 460 bytes, plus 32 bytes for each field selected (normally 864 bytes), plus 64 bytes for each field that has a description on the system database (up to 1728 bytes). A stored function shows as a field name, but does not contain a description.

The FIND and USE databases are treated as if they were two different databases, even if they are really the same database.

Q-REC-4 Holds Temporary field information.

Minimum length:	124 bytes
Maximum length:	3484 bytes

Up to 4 temporary fields can be defined per record.
Add 16 bytes for each field used as a temporary field.
Add 40 bytes for each literal used in a temporary field.

Q-REC-5 Holds qualification data.

Minimum length	460 bytes

Maximum length: 2460 bytes

Up to 36 lines of qualification held per record.
Add 56 bytes for each character literal.
Add 24 bytes for each numeric literal.

Q-REC-0 Holds vocabulary words that AQF needs and user is allowed to use. AQF always uses the shortest word with the same meaning in the vocabulary.

Minimum length: 182 bytes

Maximum length: 1798 bytes

For each vocabulary word used, normally 8 x (7+len/8) bytes of additional storage is needed. (len = length of vocabulary word). Using the default vocabulary, 462 bytes are needed.

First two bytes of TS queue is length of record. Second two bytes of TS queue is item number (key) of record. Third two bytes of TS queue is key of next item number (if records are "chained" together).

COMMAREA sizes used by VISION:Inquiry:

Native VISION:Inquiry	- 18 bytes
AQF	- 4064 bytes

AQF Input or Output Maps - II.TCVLSRC (CVLCAMOD)

The source library member II.TCVLSRC (CVLCAMOD) is provided if you want to change the mapset names or the number of help pages. You cannot change the map names inside the mapset. The source of the maps are in II.TCVLSRC library with the names IDBEHnn and IDBEMnn (nn is a two-digit number) and IDBERRS.

You must not:

1. Define the last line of a screen in a MAPSET. (Note that the error messages in IDBERRS are always put on the last line.)
2. Change the length of any input or output map fields.

3. Change, for the most part, the order input or output fields defined in a map. You can change the location in which they appear on the map (which could change the order they are seen on the map). See Rule 4 in the section below for an exception to this rule.
4. Specify EXTATT=YES (only EXTATT=MAPONLY and EXTATT=NO is supported).
5. Add or delete any input or output fields to the screen.

You can:

1. Change any text on a screen (to French, for example). Multiple languages for AQF can be supported by using the Alternate Screen Suffix, and so forth.
2. Change the location of a field.
3. Change the attributes of a field (such as bright, normal, reverse video, blue or green) within reason. Input fields are "brightened" when in error. Do not change the default to BRIGHT. For terminals that support it, highlighting fields is recommended.
4. Rearrange groups of input and output fields if you want the fields to be displayed in vertical order instead of horizontal. This is the exception to Rule (3) in the section above.
5. Change the number of help pages.
6. Take advantage of terminals whose screen size is larger than 24 by 80. However, the HELP maps must have the same number of pages (for example, IDBEH03 cannot have four pages for a 3270 terminal Model-2, but only two pages for a 3270 terminal Model-4).
7. Change the default position of the cursor on any maps. (Default is used when no errors are detected.)

AQF Batch Job Submission Considerations

The user can build the inquiry through the AQF panels and then process it either online or submit it to be processed in the batch region by the batch version of the product.

In VISION:Inquiry, all the user databases or files will be accessed for read processing only, so they can be shared by the online and batch version of the product without losing the integrity of data by using the appropriate parameters (such as shareoptions parameter). However, the system database will be updated by VISION:Inquiry for some commands, so you must consider carefully if you want to share it in online and in batch processing.

If you submit your job for batch processing through the AQF, the batch job will update the system database under the following conditions:

- When saving the inquiry
- When using generic terminal (only the first time for each terminal)
- When the mode of the terminal is defined as conversational

To protect your DL/I or VSAM system database from concurrent update in the batch and the online environments for the above conditions, you should do the following:

- If DL/I system database is used, run the batch job in a shared database environment for local DL/I under CICS or use the DBCTL facility.
- If VSAM system database is used, define the system database with shareoptions (4 3) and open with DISP=SHR. In the link edit step of the product (discussed in the *Advantage VISION:Inquiry for CICS Installation Guide*), the ENQ parameter must be coded as ENQ=Q to link the system with the modules using ENQ and DEQ macros to serialize all VSAM updates against these.

  Using the ENQ and DEQ macros might have severe impact on your CICS region response time and performance. It might stop the whole CICS region for the time the ENQ is in effect until the DEQ macro is issued.

On the AQF batch job submission panel the user has two options to build the job control statements:

- The user enters the job control statements along with the name of the procedure to be started in the PROC statement and AQF submits them without any syntax checking.
- The AQF builds and sends the job control statements using the procedure name and information entered by the user on the panel and the parameters in the member CVLCAJPR in II.TCVLSRC (refer to the *Advantage VISION:Inquiry for CICS Installation Guide*).

In both cases, the procedure should be provided to the user before using the AQF batch job submission facility.

You can use the sample JCL (IQBATD, IQBATD2, or IQBATV) in II.TCVLCNTL to build the required procedure.

Intraccess Considerations

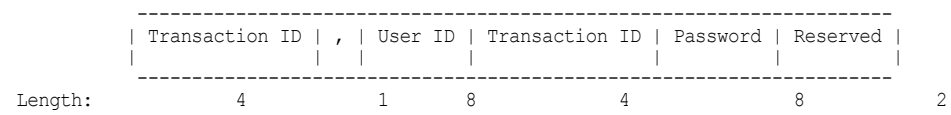
The following subsections describe VISION:Inquiry programming and operating techniques and considerations related to the execution of inquiries using the Intraccess option. For the operating considerations specific to the Intraccess programs, see the Intraccess documentation.

Online Processing

Intraccess only runs with the online version of VISION:Inquiry in the CICS region. The VISION:Inquiry queries are generated by Intraccess and passed through TCP/IP to VISION:Inquiry for processing in the CICS region. The output of the process will be saved in the CICS temporary storage and passed back to Intraccess through TCP/IP.

Any VISION:Inquiry transaction that uses the Intraccess option must have a dummy terminal defined with the name PC for one of its directories. A dummy terminal is a terminal that has the parameter PCOUT=YES (see Chapter 4, "The Definition Process") and is defined in the system database like other terminals using the IIGEN utility.

Intraccess uses the IBM-supplied TCP/IP listener to communicate with VISION:Inquiry in the CICS region. The following is the input format of the data received by the listener from client in its first transmission.



Where:

Field	Description
Transaction ID and User ID	Are equivalent to APPL name and TERM name defined in the VISION:Inquiry system database. Use the Transaction ID to start the VISION:Inquiry transaction. Use the combination of Transaction ID and User ID to get the user profile (directory) associated to the user.
Password	Is a logon password passed by Intraccess. The password can be checked by a user routine through the TCP/IP exit point provided at the Listener process. See the TCP/IP manuals for more information about the security exit point provided at the Listener process.

Programming Considerations

When the VISION:Inquiry transaction is started by Intraccess through TCP/IP, the front-end program, INQIO, gets control and receives the following data area in its first transmission from Intraccess.

	Reserved	User ID	Transaction ID	Password	Reserved	Indicator	Reserved
Length	20	8	4	8	2	10	20

Where:

Field	Description
User ID and Transaction ID	Are equivalent to TERM name and APPL name defined in the system database. They are used to get the user profile (directory) associated to the user.
Password	Is a user logon password passed by Intraccess. VISION:Inquiry does not process the password. However, it can be checked by a user routine through the TCP/IP exit point provided at the Listener process and/or the front-end INQIO program.
Indicator	Contains the word INTRACCESS and indicates that the front-end program, INQIO, is invoked by Intraccess.
Reserved	The reserved fields are the information used internally by Intraccess and VISION:Inquiry. These fields should be modified by the front-end program, INQIO.

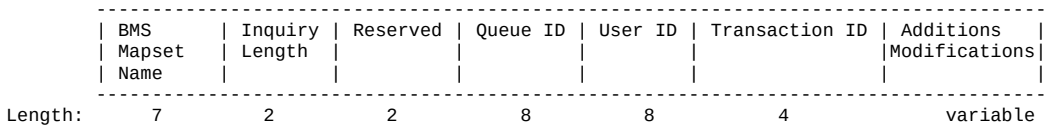
The INQIO program checks the indicator in the data area for the word INTRACCESS and passes control to the Intraccess front-end program, INQINTRA. The source of the INQIO program is available in II.TCVLSRC and can be modified or rewritten. However, the reserved fields in the data area passed by Intraccess should not be changed.

You can check and validate the password or user ID in the front-end program (INQIO). If invalid, you can terminate the query processing by putting the word **ERROR** in the indicator area in INQIO before transferring the control to the Intraccess front-end program INQINTRA. VISION:Inquiry sends an error message back to the Intraccess and terminates the process.

The INQINTRA program does the following main functions:

- Builds the I/O area used by the VISION:Inquiry main program.

The format of the I/O area is:



Where:

Field	Description
BMS Mapset Name	Is the name of the default BMS map used when the output is sent to another terminal or printer as opposed to the PC.
Inquiry Length	Is the length of the Inquiry field that follows.
Reserved	Is a field that is used internally.
Queue ID	Is the temporary storage queue ID name used to store the output before sending it to the PC.
User ID and Transaction ID	Are used to get the user profile (directory) associated to the user.
Inquiry	Is the source of the inquiry generated by Intraccess.

- Links to the VISION:Inquiry main program, IXXSTRT.
- Handles the I/O operation with Intraccess through TCP/IP.

Conversational vs. Continuous Mode with Intraccess

The concept of conversational vs. continuous mode of operation when using Intraccess is similar to that of native VISION:Inquiry (see Conversational Mode on page 6-12 and Continuous Mode on page 6-12). In continuous mode, the process is continued until its completion with no interruption. In conversational mode, the following steps will be taken:

- Processing continues until a checkpoint (either a page end or the maximum number of calls) occurs.
- The result up to this point will be sent to the PC or another printer/terminal.
- The status and position information of the inquiry will be written in the scratch pad area of the system database.
- All the resources in the CICS related to this operation will be freed.
- Processing resumes for the rest of data until another checkpoint or end of processing occurs.

Temporary Storage

VISION:Inquiry uses the temporary storage facility of CICS to store the output result before sending it to the PC when the Intraccess option is used. For each process, the Intraccess front-end program generates a unique queue ID based on the CICS task number, and the output result will be written to this queue. After the completion of the process or occurrence of a checkpoint, the temporary storage queue information will be transferred to the PC, and the queue will be deleted.

Downloading Data or a Report with Intraccess

When using the Intraccess option, either data or report can be downloaded to the PC. The data is in a tab-delimited format, and the report is either pre-formatted (non-UDO) or formatted (UDO). Based on the syntax of the inquiry, VISION:Inquiry determines whether to generate data or a report. Inquiries in UDO syntax and summary commands such as TOTAL will generate a report. Otherwise, data will be generated. The length of data can be up to 8000 bytes.

File Transfer Operation of VISION:Inquiry

File transfer involves the following:

- VISION:Inquiry extracts the data or report and writes it to the VISION:Journey download data set.
- VISION:Inquiry then starts VISION:Journey. VISION:Journey retrieves the data or report from the download data set and downloads it to a PC file.

The following sections discuss the download data set and VISION:Journey for Windows mainframe considerations. The PC considerations of VISION:Journey are explained in the *VISION:Journey for Windows System Administrator's Guide*.

VISION:Journey Online Design

VISION:Journey for Windows is a client and server system, which means the user does all interactions on the PC. The PC is the workstation client. When information is required, the PC accesses the mainframe as a data server to get requested data or report.

The VISION:Journey for Windows product use the VISION:Results, VISION:Results for DB2, VISION:Excel, and VISION:Inquiry host computer products as host server entities for the client or server relationship.

For the VISION:Inquiry product, communications between the work station client and host server are achieved using CICS or IMS. The PC side of VISION:Journey logs on to either CICS or IMS through the 3270 Presentation Space. The 3270 Presentation Space allows VISION:Journey for Windows to interact with CICS or IMS exactly the same way as a user, but with the added benefit of capturing the data on the screen in its own working storage.

Everything in VISION:Journey is processed through CICS or IMS transactions. VISION:Journey starts VISION:Inquiry through transactions as necessary for processing the inquiry, extracting the data or report, and writing it with the necessary control information to an intermediate download file. The download file is used to transfer data to the PC client.

VISION:Journey Sequence of Actions

Using CICS transactions, VISION:Journey, along with VISION:Inquiry, does the following:

1. VISION:Journey logs on to CICS.
2. VISION:Journey starts VISION:Inquiry and passes the generated query to it.
3. VISION:Inquiry executes the query and writes the result (extracted data or report) to the download file. The query contains the PCE command or OUTPUT `dummy terminal' which initiates the download process.
4. VISION:Inquiry starts the first VISION:Journey transaction (FTS1) which calls the DYLC0SS program. The FTS1 transaction sends the BMS map IDFTSP7 along with the control information to start the download.

5. VISION:Journey invokes the DYDL transaction to download the data to the PC. The DYDL transaction calls the “send/receive" program (DYLC010) which, in turn, calls the “read/write" program (DYLC020) to read the output that is to be downloaded from the download file. The “send/receive" program then calls the compress program (DYLC030) to compress the data. The compressed data is sent onto the screen, which in reality is the 3270 Presentation Space. The PC side of VISION:Journey for Windows captures the data and decompresses it. The DYDL transaction ends once all the data has been downloaded.
6. VISION:Journey flags the data for deletion in CICS in the download file.
7. VISION:Journey logs off the host.

VISION:Journey Host Computer Considerations

The System Administrator should consider the following during the installation and maintenance of VISION:Journey:

- The default BMS map INQMAP2 is required by the VISION:Journey System Administrator during the file definition process. Limited changes are allowed to this map. You can change the name of the map and the content of the protected fields but do not change the order of the fields in the maps.
- If VISION:Journey is already used for other products, such as VISION:Results, you must use the existing VISION:Journey for Windows modules. The ddname for the download file must match the ddname specified in the DJCSECT routine of the existing VISION:Journey installed for the VISION:Results product.
- Any VISION:Inquiry transaction (application) which uses the VISION:Journey facility must have a dummy terminal defined with the name PC for one of its directories. A dummy terminal is a terminal which has the parameter PCOUT=YES (see Chapter 4, “The Definition Process”) and is defined in the system database like other terminals using IIGEN utility.
- VISION:Journey uses a BMS map (supplied as IDFTSP7) at the start of the download process. You can change the name of the map, but do not change the content or the fields defined in this map.
- VISION:Journey will flag the data or report in the download file after completion of download with the status code of “D.” The cleanup utility (see Maintaining the VISION:Journey Download Data Set with IFUCLEN on page 5-52) must be run periodically to delete the downloaded data (such as “D" status codes).
- The member VSMFTSRE in control library II.TCVLCNTL is used to unload the download file to a sequential file and restore it back to eliminate the VSAM CI/CA splits and restore the unused space. Running this job periodically will improve the performance of the file.

VISION:Journey VSAM Download Data Set

The download data set is a staging file used to hold the data and reports for download processing. Each data or report is assigned a terminal or user ID along with a unique subsequence number to identify it from other data or reports.

VISION:Journey for Windows requires the download data set to be a VSAM KSDS file. The download file records consist of a 16-byte key with the size and type that comprise the key:

Terminal Name (eight characters)	VISION:Journey user or terminal identification. It can be CICS terminal ID, operator ID, or user ID. This is the same name used to define terminals to VISION:Inquiry. The type of identification is determined during VISION:Inquiry installation.
Subsequence (2-byte binary)	A unique number assigned to extracted data or report. It is used to distinguish one extracted data or report from another.
Record Type (2-byte binary)	A number used to distinguish the type of record.
Record Sequence (4-byte binary)	Consecutive number assigned to the description and data type records.
or	
Batch-key (4-byte binary)	A generated number assigned by VISION:Journey for each master header record.

The structure of the download data set is as follows:

Dummy Control Record	This is a dummy record which is used to initialize the download data set. There is only one record of this type in the data set.
Master Control Record	A record which will be created only once for each user or terminal and the first time the user or terminal executes a VISION:Journey command. It contains the next subsequence number to be assigned to a download data or report. It also contains "code points" required for transmission of data and reports between the mainframe and PC. The subsequence number of this record is zero.
Master Header Record	This record contains information required for downloading. It is created internally by VISION:Journey and is automatically deleted after the data has been downloaded.

Root Record	Root records contain information about the extracted data or report. There is one root record for each extraction.
Description Record	Description records contain information about each field of a data or report record such as its location, length, and type. The number of description records for each extraction of data is equal to the number of fields. For each extraction of report, one description is created.
Data Record	Data records are the actual data and reports to be downloaded to the PC.

The VISION:Journey download data set can also be defined to VISION:Inquiry as a user data set and its contents can be displayed using the VISION:Inquiry DISPLAY command by the System Administrator to obtain information about the extracted data or report. However, because different VISION:Journey download data set record types exist, you need to use an IF clause with the DISPLAY command to correctly display the information about each record type.

II.TCVLSRC (IIDMGEN)

The II.TCVLSRC (IIDMGEN) member contains the MAPGEN for the different record types of the download data set.

Figure 6-9 shows the MAPGEN used to define the VSAM download data set to VISION:Inquiry as a user data set. This MAPGEN does not include nor require the Master Control and the Master Header Records, since these are internally generated records.

```
*
*        MAPGEN FOR VISION:Journey VSAM DOWNLOAD DATASET
*
 MAPGEN  MAP=VSFTSMAP,FILE=VSFTSDS,NAME=VSFTS,DSTYPE=VSAMKSDS,
         DESC='FTS FEATURE/VISION:Journey VSAM DOWNLOAD DATASET'
 RECORD  BYTES=8020,TYPE=V
 FIELD   START=1,LENGTH=16,TYPE=C,NAME=VSRECKEY,KEY=SEQ-U,
         DESC='KEY OF FTS FEATURE DOWNLOAD DATA SET'
 FIELD   START=1,LENGTH=10,TYPE=C,NAME=VSGENKEY,KEYPOS=1,
         DESC='GENERIC KEY,, CONSISTS OF TERM NAME AND SUBSEQUENCE #'
 FIELD   START=1,LENGTH=8,TYPE=C,NAME=VSTERMNM,KEYPOS=1,
         DESC='TERMINAL NAME AS PART OF THE KEY'
 FIELD   START=9,LENGTH=2,TYPE=B,NAME=VSSUBSEQ,KEYPOS=9,
         DESC='SEQUENCE NUMBER AS PART OF THE KEY'
 FIELD   START=11,LENGTH=2,TYPE=B,NAME=VSRECTYP,KEYPOS=11,
         DESC='INTERNAL NUMBER INDICATING TYPE OF RECORD'
 FIELD   START=13,LENGTH=4,TYPE=B,NAME=VSRECSEQ,KEYPOS=13,
         DESC='INTERNAL RECORD SEQUENCE NUMBER'
 FIELD   START=19,LENGTH=2,TYPE=B,NAME=VSDATALN,
         DESC='LENGTH OF VARIABLE DATA'
```

Figure 6-9 MAPGEN for the VISION:Journey VSAM Download Data Set (Page 1 of 2)

```
*
*        VSFTSROOT DATA
*
 FIELD   START=21,LENGTH=8,TYPE=C,NAME=VSAPPL,
         DESC='APPLICATION NAME INITIATED THE EXTRACTION'
 FIELD   START=45,LENGTH=1,TYPE=C,NAME=VSCREST,
         DESC='STATUS OF EXTRACTED DATA'
 FIELD   START=46,LENGTH=3,TYPE=C,NAME=VSCREENV,
         DESC='FILE TRANSFER ENVIRONMENT,, CIS FOR CICS OR IMS'
 FIELD   START=49,LENGTH=4,TYPE=B,NAME=VSRCOUNT,
         DESC='NUMBER OF RECORDS EXTRACTED'
 FIELD   START=53,LENGTH=4,TYPE=P,NAME=VSCREDTE,
         DESC='CREATION DATE IN YYDDD FORMAT'
 FIELD   START=57,LENGTH=4,TYPE=P,NAME=VSCRETME,
         DESC='CREATION TIME IN HHMMSST FORMAT'
 FIELD   START=61,LENGTH=2,TYPE=B,NAME=VSCONLEN,
         DESC='PC FILE NAME LENGTH'
 FIELD   START=63,LENGTH=255,TYPE=C,NAME=VSCONST,
         DESC='PC FILE NAME'
*
*        VSFTSDESC DATA
*
 FIELD   START=21,LENGTH=2,TYPE=B,NAME=VSFLDLOC,
         DESC='STARTING LOCATION OF FIELD'
 FIELD   START=23,LENGTH=2,TYPE=B,NAME=VSFLDLEN,
         DESC='FIELD LENGTH - 1'
 FIELD   START=25,LENGTH=2,TYPE=B,NAME=VSEXLEN,
         DESC='EDITED LENGTH OF FIELD - 1'
 FIELD   START=27,LENGTH=1,TYPE=C,NAME=VSFLDTYP,
         DESC='FIELD TYPE'
 FIELD   START=29,LENGTH=2,TYPE=B,NAME=VSFLDSCL,
         DESC='FIELD SCALING FACTOR'
 FIELD   START=31,LENGTH=2,TYPE=B,NAME=VSNMELTH,
         DESC='LENGTH OF FIELD NAME'
 FIELD   START=33,LENGTH=32,TYPE=C,NAME=VSFLDNME,
         DESC='FIELD NAME'
*
*        VSFTSRECS DATA
*
 FIELD   START=21,LENGTH=50,TYPE=C,NAME=VSRECORD,
         DESC='THE FIRST 50 BYTES OF EXTRACTED RECORD'
 FINISH
```

Figure 6-9 MAPGEN for the VISION:Journey VSAM Download Data Set (Page 2 of 2)

The VISION:Journey VSAM download data set is a variable length record KSDS file with three types of records. In this section, the record types are called root record, description record, and data record. The first part of all the record types are common.

Description of the Fields

Following is the description of the fields:

VSTERMNM	Terminal name which initiated the extraction
VSSUBSEQ	Unique subsequence number assigned to the extraction
VSRECTYP	Type of record; for example, 0 for root; 1 for description; and 2 for data record type
VSRECSEQ	The internal record sequence number for description and data record types (starting from 1). It is always zero for root record type.
VSDATALN	Length of the remaining record

Variable Part of the Root Record Type

The variable part of the root record type contains general information about each extraction. There is one root record for each extraction.

VSAPPL	Application name
VSCREST	Extracted data status: F=completed C=checkpoint taken I=incomplete due to some errors D=flagged for deletion (VISION:Journey for Windows only)
VSCREENV	Environment (CIS for CICS version)
VSRCOUNT	Number of data records extracted
VSCREDTE	Creation date (YYYYDDD)
VSCRETME	Creation time (HHMMSST)
VSCONLEN	File name length
VSCONST	PC file name; for example, the name specified as a literal following the VISION:Inquiry command used for file transfer

Variable Part of the Description Record Type

The variable part of the description record type contains the information about each field to be extracted. The number of description records for each extraction is equal to the number of fields to be extracted.

VSFLDLOC	Starting location of field within the record
VSFLDLEN	Field length — 1
VSEXLEN	Output length of the field — 1. The output length is the length specified in the OUTLTH parameter in the field statement of MAPGEN (see Chapter 4, "The Definition Process").
VSFLDTYP	Type of field. The same as the type parameter used in the field statement of MAPGEN (see Chapter 4, "The Definition Process").
VSFLDSCL	Scaling factor. The same as the scale parameter used in the field statement of MAPGEN (see Chapter 4, "The Definition Process").
VSNMELTH	The length of the field name
VSFLDNME	Field name

Variable Part of the Data Record Type

The variable part of the data record type contains actual data extracted from the user database or file. The number of data record types is equal to the number of data records for each extraction.

VSRECORD	Data record. The data record can be up to 8000 bytes. In the above example, the record shows only the first 50 bytes of the record.

Native SQL Syntax Facility

Native SQL syntax is a facility of native VISION:Inquiry which can be invoked in four environments and two modes of operation supported by native VISION:Inquiry.

The considerations discussed previously in Environments and Environment Processors on page 6-1 through Online Extraction with the EXTRACT Command on page 6-34 is also applicable. This facility allows SQL SELECT statements combined with some VISION:Inquiry commands to be processed against DB2 tables or views and the output returned to the terminal.

Native SQL Syntax considerations

The following considerations apply to the Native SQL syntax facility:

- The keywords, EXECSQL and ENDEXEC, are reserved words used only as SQL statement delimiters in the inquiry.
- The information regarding DB2 tables or views accessed by native SQL syntax facility will be extracted from the DB2 catalog directly and no MAPGEN is required to define the tables or views in the system database.
- The output will be returned in non-udo format in either vertical or horizontal format with the column headings the same as the column names returned in the SQLDA by DB2. In case of no name such as expressions, a period will be displayed as column heading.
- The length of the edited column values in the output report will depend on the type of the column:

Column Type	Output length
CHAR	n (n = # of characters)
VARCHAR	n (n = max. # of characters)
DECIMAL	n (Depends on precision and scale)
SMALLINT	7
INTEGER	14
DATE	10
TIME	8
TIMESTAMP	26

Column Type	Output length
REAL(Floating point)	16 (Valid if output exit is used)
DOUBLE(Floating point)	23 (Valid if output exit is used)
GRAPHIC	n*2 (Valid if output exit is used, n = # of DBCS)
VARGRAPHIC	n*2 (Valid if output exit is used, n = max. # of DBCS)

- The numeric values in the output report will be edited with zero suppression, decimal and negative sign placement, and comma insertion where applicable for example, 123,45.78-.
- The null value is displayed as "?" in the output report for example, for a character column with the length of 4 the null value is displayed as ????. However through the output user exit (discussed later in this guide), you can specify what to be displayed for the null value.
- The null value will be shown as X'00' in the extract data set.
- The floating point and graphic values can be processed by the output user exit only (discussed later in this guide). If no output exit is used, VISION:Inquiry will issue an error message and processing terminates.
- If the VISION:Inquiry LIMIT command is used with the DB2 ORDER BY command, the limit applies after the output is sorted.
- Can be used with the EXTRACT command with the same considerations discussed in Online Extraction with the EXTRACT Command on page 6-34 through Extract Data Set Format on page 6-34 and Batch Extraction with the EXTRACT Command on page 6-31 through Extract Data Set Format on page 6-32.
- Can be used with the Text Editor facility with the same considerations discussed in Text Editor Map Definition Specifications on page 6-15.
- The inquiries will be stored in the system database in source format only and can then be edited, executed, displayed, or deleted using appropriate VISION:Inquiry commands.
- The SQL command "ORDER BY" can be used to sort the data.
- Not supported in AQF mode.
- Not supported with VISION:Journey for Windows.

7 User Exits

User exits are logical points in the VISION:Inquiry system flow where you can interface to VISION:Inquiry with your own programs. Through these exits you can provide functions that are not currently included in the VISION:Inquiry system.

Note: This chapter contains information for sites licensed with the VISION:Inquiry DB2 option and without the DB2 option. Text containing "DB2" is specifically applicable to DB2 licensed sites.

Introducing the User Exits

VISION:Inquiry provides the following user exits:

IXXUIN input exit	This exit modifies inquiry statements before they are processed by VISION:Inquiry. It is commonly used to add conditional selection clauses to inquiries to limit access to specific data from certain terminals. This exit can also be used to change the specifications for the system database and the connection to DB2.
IXXUOUT output exit	This exit modifies or deletes output fields in the output of an inquiry. It is commonly used to provide value security to the inquiry or to change the output characteristics of a field.
IXXUCON conversion exit	This exit converts input, output, and internal fields whose type was specified in the MAPGEN as TYPE=U or TYPE=0-9. This allows you to present encoded data to the user in decoded form or to process data types (for example, calendar dates or floating point fields) not directly supported by VISION:Inquiry
IXXUFNC function exit	This exit is called when a user function is used in an inquiry. It can be used to implement arithmetic computations beyond the scope of ordinary VISION:Inquiry processing.

IXXUSEC security exit	This exit provides for security which is not based on the standard mechanisms, transaction code and terminal name. This exit can be used for native mode and for AQF.
IXXUVSM VSAM exit	This exit changes or expands the VSAM record read by VISION:Inquiry to match with its MAPGEN definition for variable length records.
IXCUEXT extract exit	This exit is called for each EXTRACT command to be processed by VISION:Inquiry in the online environment. VISION:Inquiry then passes the destination ID of the extract data set to the exit routine. Using this exit, you can change the destination ID. It is commonly used to assign different destination IDs to different users as opposed to a single destination ID (DYL1).
IXCUSRT sort exit	This exit is called at the end of each online sort processing when the VSAM work data set technique is selected in the CICS parameters module at the installation time. The exit routine can be used to start a new transaction to delete the output records from the VSAM work data set after the completion of sort. It is used to improve sort response time when the VSAM work data set technique is used.
IXCUOUT CICS output handling exit	This exit is called at the output processing stage to handle the output report. It bypasses the standard CICS handling of VISION:Inquiry output processing which uses the CICS BMS paging facility. ■ This exit is not used with the VISION:Journey facility. ■ It is only used with the Intraccess option when sending the output report to another terminal or printer.

IXXUIN, IXXUSEC, IXXUOUT, and IXCUOUT are the only user exits available for the native SQL syntax facility of VISION:Inquiry.

The output exit, IXXUOUT, is the only exit which differs when invoked by native SQL. The detail description of its operation will be discussed later in this chapter.

To assist you in developing user exits appropriate to your installation, examples of the input, conversion, and function exits are given in Appendix B, "Sample User Exits". These examples are provided on an as-is basis with no warranty from Computer Associates.

Programming and Installation Considerations

User exits can be written in either PL/I or Assembler. Other languages also may be used provided they can be interfaced following PL/I subroutine conventions.

To preserve re-entrance in the inquiry programs, user exits must also be re-entrant. Since user written exits are included in the VISION:Inquiry load modules, the amount of storage required to execute inquiries is affected.

User exits must maintain the integrity of the VISION:Inquiry system. This includes preserving the PL/I environment in which the inquiry programs operate. Considerations for exits written in Assembler and in PL/I are discussed in Assembler Exit Considerations on page 7-7, and PL/I Exit Considerations on page 7-11. A detailed discussion of the PL/I environment and the constraints imposed on Assembler language programs can be found in the IBM publication PL/I for MVS and VM Programmer's Guide.

User Exits are incorporated into the VISION:Inquiry system under the control of SMP/E and as a USERMOD.

After an exit has been assembled or compiled, you should keep the object module. You should then build an SMP/E USERMOD, include the user exit object, and run the SMP/E RECEIVE, APPLY, and ACCEPT to incorporate the user exit into your load modules.

Figure 7-1 on page 7-4 shows an example of the USERMOD that can be used as a model to add the input user exit, IXXUIN, to your system.

You can use the following model JCL members provided in the II.PREP.CNTL data set to RECEIVE, APPLY, RESTORE, and ACCEPT the USERMODs. At most sites, there are ISPF-driven facilities that can also be used to perform these SMP/E processes.

JCL Member	Description
IQSMPE#A	RECEIVE a USERMOD into the Global Zone/Data sets.
IQSMPE#B	APPLY a USERMOD to the target libraries.
IQSMPE#C	ACCEPT a USERMOD to the distribution libraries.
IQSMPE#D	RESTORE (remove) a USERMOD from the target libraries.

Note: Once you ACCEPT an element, such as a USERMOD into the distribution libraries, there is no direct method for restoring the previous version of an element.

```
++USERMOD(IQUIN001) /* USER INPUT EXIT
  THIS A SAMPLE USER INPUT EXIT, IXXUIN
        */.
++VER(Z038) FMID(CCVL650).
++MOD(IXXUIN) DISTLIB(INQOBJD).
.
.
.
Your input user exit object deck will go here
.
.
.
```

Figure 7-1 Sample USERMOD for the User Exits

Each VISION:Inquiry user exit has a single entry point. If, for example, you want to implement user conversion types for calendar date fields and clock time fields, you must design one exit with an entry point named IXXUFNC and determine, within the exit, which type of field is being processed. By using the various items of information passed to each exit you should be able to control your user exit's processing to any desired degree.

Communication Between VISION:Inquiry and User Exits

User exits and VISION:Inquiry communicate by passing parameters, issuing messages, and calling entry points.

Passing Parameters

VISION:Inquiry passes parameters to the user exits. The parameter list for each exit is different; however, the first two parameters are always the same (except for IXXUVSM, IXCUOUT, and IXCUEXT):

- The first parameter is the list of PCB addresses from the PSB passed to VISION:Inquiry by IMS (DL/I). In batch, if the inquiry program is not executing under IMS (DL/I), this parameter is the address of the parameter list passed to VISION:Inquiry.
- The second parameter, called the exit indicator, contains the following fields important to your exits.

Offset	Length	Format	Content
0	1	Bit	Type of exit: X'80' = input exit X'40' = output exit X'20' = conversion exit X'10' = function exit X'08' = security exit
1	1	Bit	Type of call for output, conversion, and security exits (see individual exit descriptions)
2	1	Bit	Return code for output, conversion, and security exits (see individual exit descriptions)
3	1	Char	Calling inquiry program: B = batch C = CICS
4	4	Char	DB2 subsystem ID if CALL Attach is used; otherwise blanks
8	8	Char	DB2 plan name if CALL Attach is used; otherwise blanks
16	4	Char	System database type: IMS, DB2, or VSAM
20	4	Addr	For IMS (DL/I) system database: address of PCB
20	27	Char	For DB2 system database: authid.tablename

Note: Since the addressing mode specification of the VISION:Inquiry load modules are 31, the parameters passed to the user exits should be treated as 31 values and addresses.

The parameter for IXXUVSM, IXCUOUT, IXCUEXT, and the remaining parameters for the other user exits are discussed later in this chapter.

Issuing Messages

User exits can issue error messages. Error message numbers 8001-8999 are reserved for user messages, which may be stored either in the system database or the hard-coded messages module. (If the same message number is stored in both places, the hard-coded message takes precedence.)

Store your messages in the same way as the supplied VISION:Inquiry messages:

- For system database resident messages, see Chapter 5, "The Utilities", for defining messages for native VISION:Inquiry.
- For hard-coded messages, see the *Advantage VISION:Inquiry for CICS Installation Guide*, Chapter 2 "Installation," for customizing hard-coded messages.

You may also use existing VISION:Inquiry messages, or message number 0, which indicates the message to be issued consists of only the substitution text.

Entry Points and Messages

There are three entry points within the inquiry processing programs that you can call to issue VISION:Inquiry messages from a user exit:

- IXSERRA, for issuing messages with substitutable text from PL/I or Assembler routines, requires three parameters: a full-word containing the message number, a pointer to the substitution text, and a half-word containing the length of the substitution text (or zero).
- IXSERRB, for issuing messages without substitutable text, requires one parameter: a halfword containing the message number.
- IXSERRC, for issuing messages with substitutable text from PL/I routines, requires two parameters: a halfword containing the message number and the substitution text (or the null string).

The appropriate PL/I declarations for these routines are:

```
DCL IXSERRA EXTERNAL ENTRY ( BIN(15,0), PTR, BIN(15,0) );
DCL IXSERRB EXTERNAL ENTRY ( BIN(15,0) );
DCL IXSERRC EXTERNAL ENTRY ( BIN(15,0), CHAR(*) );
```

If a user input exit is to change the specifications for the system database or for the DB2 connection when using a DB2 system database, it must not issue any error messages except those contained in the hard-coded messages module or message number 0.

Assembler Exit Considerations

Exits written in Assembler must establish the PL/I environment and maintain proper prolog and epilog conventions. Two Assembler language macros are provided in order to maintain these requirements. These macros are IXBEGIN, which is coded at the beginning of your Assembler code, and IXRETURN, which is coded at the end of your Assembler code.

If your Assembler exit routine needs to use some area in DSA (Dynamic Storage Area), you should specify the length parameter in the IXBEGIN macro. The default length of DSA is 104 which is used as a register save area. Figure 7-3 on page 7-9 shows how you may define a 20-byte area in DSA to be used in your user exit.

If your Assembler exit routine uses CICS commands, the IXBEGIN macro should specify CICS=YES as a parameter. This includes the DFHEIENT macro for CICS prologue and establishes the PL/I environment. In this case, Register 2 is a program base register; Register 8 points to the start of the CICS dynamic area (such as DFHEISTG), and Register 10, where applicable, points to the start of the Exec Interface Block (EIB).

IXUSER Macro

An additional macro, IXUSER, is provided for your convenience. It contains Assembler descriptions of the individual VISION:Inquiry data elements used in the exits. Figure 7-2 contains the source code for the IXUSER macro. Note that this macro cannot be used for native SQL syntax inquiries.

```
         MACRO
         IXUSER
*
*               VISION:Inquiry USER EXIT INTERNAL DESCRIPTIONS
*
*               THIS MACRO PROVIDES USER EXITS WITH THE DSECT
*               DEFINITIONS OF INTERNAL DESCRIPTORS
*
*
*               SEGMENT DESCRIPTOR
*
SD       DSECT
SDFUNC   DS     H     CODE X'7102'
SDLTH    DS     H     SEGMENT DESCRIPTOR LENGTH
SDUP     DS     H     UP OFFSET (PARENT)
SDFLDL   DS     H     KEY FIELD LENGTH
SDPATH   DS     H     KEY PATH LENGTH
```

Figure 7-2 IXUSER Macro Source Code (Page 1 of 3)

```
SDSEGL   DS     H     SEGMENT LENGTH
SDVALL   DS     H     MAX SSA VALUE LENGTH
SDSEGN   DS     XL1   SEGMENT LEVEL
SDIND    DS     XL1   INDICATOR
SDCOUNT  DS     F     POSITION COUNTER
SDKEY    DS     CL8   SEGMENT KEY FIELD NAME
SDSEGNME DS     CL8   SEGMENT NAME
SDCOMM   DS     CL2   COMMAND CODE
SDLP     DS     CL1   LEFT PARAN
SDFIELD  DS     CL8   SEARCH FIELD NAME
SDOPER   DS     CL2   OPERATOR
SDVALUE  DS     CL255 KEY VALUE MAX=255
SDRP     DS     CL1   RIGHT PARAN
*
*              BIT SETTINGS FOR SDIND
*
SDKEYF   EQU    X'80' SEGMENT IS KEYED
SKPATHF  EQU    X'40' SEGMENT IS QUALIFIED PATH
SDSELF   EQU    X'20' SEGMENT IS SELECTED
SDQUALF  EQU    X'10' SEGMENT IS QUALIFIED
SDFQUALF EQU    X'08' QUALIFY EQUAL ONLY
SDINDXF  EQU    X'04' FIELD IS INDEX FIELD
SDINDVF  EQU    X'02' VARIABLE LENGTH SEGMENT
SDINDFLF EQU    X'01' FIRST OR LAST QUALIFICATION
*
*              DATA DESCRIPTOR
*
DD       DSECT
DDFUNC   DS     H  CODE  X'7000' ROOT KEY DESCRIPTOR
*                       X'7001' KEY FIELD DESCRIPTOR
*                       X'7002' KEY SYNONYM DESCRIPTOR
*                       X'7003' INDEX FIELD DESCRIPTOR
*                       X'7004' SEARCH FIELD DESCRIPTOR
*                       X'7008' NON-KEY FIELD DESCRIPTOR
*                       X'7009' NON-KEY SYNONYM DESCRIPTOR
DDLTH    DS     H   DATA DESCRIPTOR LENGTH
DDSEG    DS     H   DATA SEGMENT DESCRIPTOR OFFSET
DDFLDO   DS     H   DATA START OFFSET - 1
DDFLDT   DS    XL1  DATA TYPE
DDFLDE   DS    XL1  DATA EDIT TYPE
DDFLDL   DS    XL1  DATA LENGTH - 1
DDFEDL   DS    XL1  DATA EDITED LENGTH - 1
DDFLDS   DS    XL1  DATA SCALE
DDSUBO   DS    XL1  DATA SUB-FIELD START
DDSUBL   DS    XL1  DATA SUB-FIELD LENGTH
DDMAPID  DS    CL1  DATA MAP ID
DDMAMEL  DS    H    DATA NAME LENGTH - 1
DDNAME   DS    CL32 DATA NAME VARIABLE LENGTH MAX(32)
DDIMSNME DS    CL8  IMS SEARCH FIELD NAME
*
*              FUNCTION DESCRIPTOR
*
FD       DSECT
FDFUNC   DS    H    CODE   X'7201'
FDLTH    DS    H    FUNCTION DESCRIPTOR LENGTH
FDLISTO  DS    H    FUNCTION LIST ARRAY OFFSET
FDVALO   DS    H    FUNCTION VALUE ARRAY OFFSET
FDFLDT   DS    XL1  FUNCTION DATA TYPE
FDFLDE   DS    XL1  FUNCTION EDIT TYPE
FDFLDL   DS    XL1  FUNCTION DATA LENGTH -1
FDFLDEL  DS    XL1  FUNCTION OUTPUT EDIT LENGTH - 1
FDFLDS   DS    XL1  FUNCTION SCALE
FDLISTN  DS    XL1  NUMBER OF OPERATION LIST ENTRIES
FDVALN   DS    XL1  NUMBER OF VALUE ENTRIES
FDMAPI   DS    XL1  FUNCTION MAP ID
FDNAMEL  DS    H    FUNCTION NAME LENGTH -1
FDNAME   DS    CL32 FUNCTION NAME VARIABLE LENGTH MAX(32)
*
*              FUNCTION OPERATION ARRAY
*
FDLIST   DSECT
FDOP     DS    H    OPERATION CODE
FDOPER   DS    H    OPERAND
```

Figure 7-2 IXUSER Macro Source Code (Page 2 of 3)

```
*
*              FUNCTION VALUE ARRAY
*
FDVALUE  DSECT
FDVOFF   DS    H    DATA DESCRIPTOR OFFSET
FDVFLAG  DS    XL1  FLAG
FDVLTH   DS    XL1  VALUE LENGTH - 1
FDVVAL   DS    CL255 VARIABLE LENGTH MAX(255)
         MEND
```

Figure 7-2 IXUSER Macro Source Code (Page 3 of 3)

Notes:

- The Assembler routine you write must not alter register 12.
- The IXBEGIN macro initial base register is 2.
- Register 4 contains the address of the input parameter list.
- If the Assembler routine you write uses CICS commands, in addition to registers 2, 4, and 12, register 8 contains the address of the CICS dynamic storage area, DFHEISTG.

Example Assembler Input Exit - CICS Commands Not Used

The LENGTH parameter in the IXBEGIN macro and DSA DSECT in Figure 7-3 are only needed if you are going to define and use some area in DSA (for example, USERAREA).

Figure 7-3 illustrates the basic skeleton of the Assembler input exit when CICS commands are not used in the exit routine.

```
IXXUIN      IXBEGIN TYPE=USER,LENGTH=124
*
*
*
            USING PARMLIST,R4          ADDRESSABILITY FOR PARMLIST
            USING DSA,R13              ADDRESSABILITY FOR DSA
*
*           REG2 IS THE BASE REGISTER
*           REG12 MUST NOT BE ALTERED
*
*********************************************************************
*                                                                   *
*                 ADDITIONAL CODE CAN BE APPENDED HERE              *
*                                                                   *
*                  PLEASE BEWARE OF DUPLICATE LABELS                *
*                                                                   *
*********************************************************************
*
*
*  RETURN TO CALLER
*
*
            IXRETURN
*
*
            LTORG
*
```

Figure 7-3 Example of a Basic Skeleton of an Assembler Input Exit (Page 1 of 2)

```
*
PARMLIST  DSECT
APSB      DS     A                    ADDRESS OF PSB
AEXIT     DS     A                    ADDRESS OF EXIT INDICATOR WORD
AINPUT    DS     A                    ADDRESS OF MESSAGE INPUT AREA
ATRANCODE DS     A                    ADDRESS OF TRANSACTION NAME
ALTERM    DS     A                    ADDRESS OF TERM NAME
*
DSA       DSECT
          DS     26F
USERAREA  DS     CL20
*
          IXUSER
*
          END
```

Figure 7-3 Example of a Basic Skeleton of an Assembler Input Exit (Page 2 of 2)

Example Assembler Input Exit - CICS Commands Used

Figure 7-4 is an example of a basic skeleton of an Assembler input exit when CICS commands are used.

```
*ASM XOPTS(NOEPILOG,CICS)
DFHEISTG          DSECT
                  DFHEISTG
IXXUIN            IXBEGIN TYPE=USER,CICS=YES
*
*
                  USING PARMLIST,R4 ADDRESSABILITY FOR PARMLIST
*
*
*                 REG2 IS THE BASE REGISTER
*                 REG8 POINTS TO CICS DYNAMIC AREA, DHFEISTG
*                 REG12 MUST NOT BE ALTERED
*
************************************************************
*                                                          *
*         ADDITIONAL CODE CAN BE APPENDED HERE             *
*                                                          *
************************************************************
*
*    RETURN TO CALLER
*
                  DFHEIRET
*
                  LTORG
*
*
PARMLIST          DSECT
APSB              DS A ADDRESS OF PSB
AEXIT             DS A ADDRESS OF EXIT INDICATOR WORD
AINPUT            DS A ADDRESS OF MESSAGE INPUT AREA
ATRANCODE         DS A ADDRESS OF TRANSACTION NAME
ALTERM            DS A ADDRESS OF TERM NAME
*
*
                  IXUSER
*
                  END
```

Figure 7-4 Example Skeleton of an Assembler Input Exit containing CICS Commands

PL/I Exit Considerations

Exits written in PL/I must use the PL/I conventions for defining addresses for the passing of parameters from VISION:Inquiry to the exit routine. Each parameter is passed as a storage address. Your exit, in order to obtain these storage locations, must define the parameters as pointers.

Figure 7-5 illustrates a basic skeleton of a PL/I input exit.

Example PL/I Input Exit

For additional information, refer to the subroutine considerations discussion in the *IBM PL/I for MVS and VM Programmer's Reference Guide*.

```
IXXUIN:   /* SAMPLE USER INPUT EXIT */
          /* THIS SAMPLE EXIT ILLUSTRATES THE TECHNIQUE RECOMMENDED   */
          /* IN THE PL/I PROGRAMMER'S GUIDE FOR USING PARAMETERS      */
          /* PASSED FROM AN ASSEMBLER-LANGUAGE PROGRAM  THE PARA-     */
          /* METERS ARE DECLARED AS ARITHMETIC TO AVOID THE USE OF    */
          /* LOCATOR/DESCRIPTORS.                                     */
          PROC (I1,I2,I3,I4,I5);
          DCL  (I1,I2,I3,I4,I5)            FIXED BIN,
          (P1,P2,P3,P4,P5)                 PTR,
          PSB_LIST (15) BASED (P1)         PTR,
          1 EXIT INDICATOR BASED (P2),
                2 BYTE (4)                 BIT (8) ALIGNED,
          1 INPUT_MESSAGE BASED (P3),
                2 LTH                      FIXED BIN,
                2 LLZZ                     CHAR (2),
                2 TEXT                     CHAR (2040,REFER (LTH)),
          TRANCODE BASED (P4)              CHAR (8),
          LTERM BASED (P5)                 CHAR (8),
          ADDR BUILTIN;
          P1 = ADDR (I1);
          P2 = ADDR (I2);
          P3 = ADDR (I3);
          P4 = ADDR (I4);
          P5 = ADDR (I5);
          /* TEST EXIT INDICATION FOR INPUT EXIT */
          IF EXIT INDICATOR,BYTE(1) ¬= '10000000'B
                THEN RETURN;
          /* AT THIS POINT ALL VARIABLES ARE SET UP FOR PROCESSING    */
          END IXXUIN;
```

Figure 7-5 Example of a Basic Skeleton of a PL/I Input Exit

Input Exit IXXUIN

IXXUIN, the input exit:

- Validates and modifies the inquiry statements.
- Sets specifications for the system database or the connection to DB2.

To do this, the exit places the desired specifications into the appropriate fields of the exit indicator before returning to VISION:Inquiry, which then uses the modified values.

Use

The most common use of this type of exit is to provide additional security. The exit can be used for the following:

- The transaction code or terminal name can be checked against a table.
- A conditional selection clause that provides value security based on the input terminal can be added.
- A password contained in the statement can be checked against a password table.

Invoking the Exit

The exit is called once per inquiry. Therefore, when using stored inquiries, the exit is invoked before the inquiry is retrieved. This must be taken into consideration when writing the exit because the result could be erroneous inquiries.

The module name for this exit must be IXXUIN.

IXXUIN Parameters

The parameter list passed to the exit is:

OFFSET	CONTENT
0	PSB
4	EXIT INDICATOR
8	INPUT
12	TRANSACTION
16	TERM

IXXUIN Parameters Explanations

PSB — The address of the PCB address list passed to VISION:Inquiry when the PSB was scheduled.

EXIT INDICATOR — The address of the exit indicator.

INPUT — The address of the message input area.

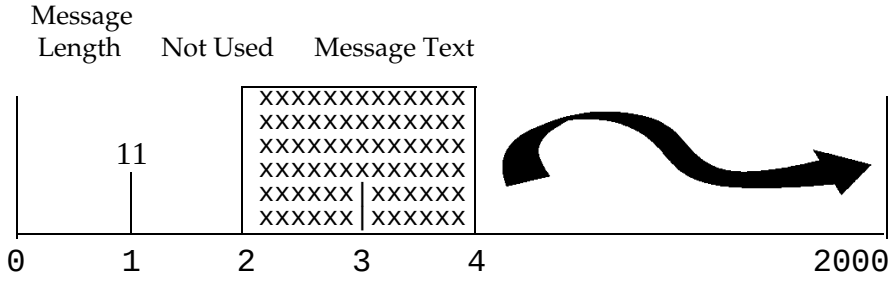
- The first two bytes of the message input area contain the length of the message.
- Bytes 5 through 2000 contain the message text.

Semicolons at the end of the message text are still physically present but are not reflected in the message length.

If the exit changes the length of the message text by adding conditional selection or by removing a password, it must also change the length indicator in the first two bytes to reflect the change.

Your exit is entered the last time it is used with a length setting of zero. When this occurs, your exit must be coded so that an immediate exit is performed.

If the length of the message is set to zero by the exit, the inquiry is ignored and VISION:Inquiry assumes that all input has been processed.



TRANSACTION — The address of the 8-byte transaction name.

TERM — The address of the 8-byte terminal name.

Output Exit IXXUOUT

IXXUOUT, the output exit, modifies or deletes the output fields of an inquiry.

The output exit is used differently for inquiries with or without native SQL SELECT statements. By checking the content of the address in offset 20 of the passed parameter list, the exit can determine whether or not the inquiry contains a native SQL SELECT statement.

- If the segment name contains the reserved word, EXECSQL, then the inquiry contains the SQL SELECT statement. Use IXXUOUT for coding the exit as described in section IXXUOUT for Inquiries with Native SQL Statement on page 7-18.
- Otherwise use IXXUOUT as described in section IXXUOUT for Inquiries with no Native SQL Statement on page 7-14.

The following sections describe both cases in detail.

IXXUOUT for Inquiries with no Native SQL Statement

When the exit receives control, it is passed the address of the source field that contains data from a database/VSAM field or the value of a variable.

The most common usage of this exit is to provide additional security based upon the value of the field or variable or to change the output characteristics of a field. The results of the exit processing are placed in a target field and the address of this field is returned to VISION:Inquiry.

- The output exit is entered during the initialization of the heading line. If the exit sets Byte 3 of the exit indicator to X'80', the heading line is not produced.
- Subsequent entries into the exit occur for each field or prior to their being output. If the exit sets Byte 3 of the exit indicator to X'20', the field is not processed by VISION:Inquiry and the data in the target field is used.

VISION:Inquiry passes control to the output exit depending on whether or not UDO is being used.

Non-UDO Inquiry

Non-UDO inquiry control is passed under the following circumstances:

- The output exit is entered once during initialization of the heading line. If the exit sets Byte 3 of the exit indicator to X'80', the heading line is not produced.
- Subsequent entries into the exit occur for each field prior to its being output. If Byte 3 of the exit indicator is set to X'20', the source field is not processed by VISION:Inquiry; the data placed in the target field by the user exit is used instead.
- Entries into the exit also occur for the control fields of subtotal commands prior to their being output.

The output exit is not called for the SUBTOTALS.

UDO Inquiry

UDO Inquiry entries into the exit occur for each database/VSAM field or temporary field prior to its being output. If the exit sets Byte 3 of the exit indicator to X'20', the source field is not processed by VISION:Inquiry and the data in the target field is used.

With a UDO inquiry, this is the only time that VISION:Inquiry passes control to the output exit.

The module name must be IXXUOUT.

IXXUOUT Parameters

The input parameter list is:

OFFSET	CONTENT
0	PSB
4	EXIT INDICATOR
8	SOURCE
12	TARGET
16	DATA DESCRIPTOR
20	SEGMENT NAME
24	DATABASE/VSAM FILE NAME

IXXUOUT Parameter Explanations

PSB

The address of the PCB address list passed to VISION:Inquiry when the PSB was scheduled.

EXIT INDICATOR

The address of the exit indicator.

Byte 2 (Call type):

- X'40' Called for headings (non-UDO only)
- X'20' Called for field; SORT was not used
- X'10' Called for field; SORT was used

Byte 3 (Return code):

One of the following values should be set in the return code field of the exit indicator.

During the initialization of the heading line (non-UDO only):

- X'80' Heading line is not produced if the bit is on.
- X'00' Headings are produced.

During field calls:

- X'20' Indicates that the source field is not processed by VISION:Inquiry and the data placed in the target area is used for DISPLAY.
- X'80' Indicates that an error condition occurred during the field conversion exit. Question marks (??) are displayed for the field.
- X'00' Indicates that the normal VISION:Inquiry DISPLAY is to occur for this field.

SOURCE

The address of the data or temporary field in the segment. The length of this field is governed by DDFLDL (which is the field length minus one).

TARGET

The address of a variable area (the target area) with a maximum length of 255 bytes where the exit must place the result of the action.

The length of the data that is placed in this field is governed by DDFEDL (which is the edited length minus one) in the data descriptor.

The edited length is either calculated by the utility program IIGEN (from the data type and length) or overridden by the OUTLTH parameter on the field statement.

The length of the exit result must be within this edited length.

Upon return, VISION:Inquiry uses this length to format the output line. If the length of the exit result is less than the edited length, it must be padded with blanks.

- You must place the results into the target area starting at the second position.
- The first position of the target area is reserved for use by VISION:Inquiry.

DATA DESCRIPTOR

The address of the data descriptor for this field.

SEGMENT NAME

The content of the segment name depends on the type of file to which the exit is referring.

- For VSAM non-hierarchical data sets, it is the address of the 3-byte field that contains the character "," followed by the type of VSAM file:

  ",KS" for KSDS
  ",ES" for ESDS
  ",RR" for RRDS

- For IMS (DL/I) databases and VSAM hierarchical data sets, the content of the segment name is the address of an 8-byte segment name. This is the name of the most recently retrieved segment, not necessarily the segment containing the field being converted.
- For DB2 tables, the content of the segment name is the address of an 8-byte field containing the word "DB2SEG".

Note that if the segment name contains the reserved word, EXECSQL, then the inquiry contains the SQL SELECT statement. Use IXXUOUT as described in the section IXXUOUT for Inquiries with Native SQL Statement on page 7-18 for coding the exit.

DATABASE/ VSAM FILE NAME

The address of the 8-byte database/VSAM file name.

The output exit can be used to modify the format of a field for display purposes. In the output exit parameters, the SOURCE parameter address points to the data field and the DATA DESCRIPTOR parameter points to a table describing the fields.

- If a user wishes to modify the field in any way, the output exit routine should be used to extract the field from the source area and edit it as desired, placing the result in the target area.
- The SOURCE and DATA DESCRIPTOR tables are for reference only. The output exit should not modify them in any way.

IXXUOUT Output

In the user output exit, the TARGET parameter address holds the resulting data after the exit has manipulated it.

- The first position of TARGET is for the target data length minus one.
- The actual data should start in the second position of the target area. Currently VISION:Inquiry does not use the length in the target area, but instead uses the length in the data descriptor field DDFEDL. The target data, however, must still start in the second position.

IXXUOUT for Inquiries with Native SQL Statement

When the exit receives control, it is passed the address of the source column in SQLDA that contains the characteristics and data from the DB2 table.

Use

The most common usage of this exit is to provide additional security based upon the value of the column or to change the output characteristics of the field. The result of the exit's processing are placed in a target field and the address of this field is returned to VISION:Inquiry.

Invoking the Exit

The exit is entered during the initialization of the heading line. If the exit sets byte 3 of the exit indicator to X'80', the heading line is not produced. The default length of the edited output column(s) (described in Chapter 6, "Programming and Operation Considerations") can also be decreased at initialization for column format processing only.

Subsequent entries into the exit occur for each column prior to its being output. If the exit sets Byte 3 of the exit indicator to X'20', the column is not processed by VISION:Inquiry and the data in the target field is used. The floating point and graphic types of data are supported through the output exit only.

IXXUOUT Parameters

The input parameter list is as follows:

OFFSET	CONTENT
0	PSB
4	EXIT INDICATOR
8	SOURCE DATA DESCRIPTOR
12	TARGET
16	EDITED OUTPUT
20	SEGMENT NAME
24	TABLE NAME

IXXUOUT Parameter Explanations

PSB — The address of the PCB address list passed to VISION:Inquiry when the PSB was scheduled.

EXIT INDICATOR — The address of the exit indicator. The first three bytes of the exit indicator are used for SQL syntax inquiries which contain:

Byte 1 (exit type):

- X'40' Indicates it is output exit

Byte 2 (Call type):

- X'40' Initialization call; Called for headings and to modify the default length of the output column (column format only).
- X'20' Column call; Called for column when "ORDER BY" was not used.
- X'10' Column call; Called for column when "ORDER BY" was used.

Byte 3 (Return code):

One of the following values should be set in the return code field of the exit indicator.

During initialization call:

- X'80' Heading line is not produced (column format only).
- X'00' Headings are produced.

During column calls:

- X'20' Indicates that the source column is not processed by VISION:Inquiry and the data placed in the target area is used for DISPLAY.
- X'80' Indicates that an error condition occurred during the exit processing. Question marks (??) are displayed for the field.
- X'00' Indicates that the normal VISION:Inquiry DISPLAY is to occur for this column.

SOURCE DATA DESCRIPTOR

During initialization calls:

The SQLDA address. The exit routine can not modify any of the SQLDA fields.

During column calls:

The address of the occurrence of SQLVAR in SQLDA for the column in process. The exit routine can not modify any of the SQLVAR fields.

TARGET

This parameter is only used during column calls:

- For columns with the edited length of more than 254 bytes (that is, long string column types), the address of the source data area. The exit must replace the source data with the result of action. For variable data types, the target data should not contain the two byte length prefix and the length will be taken from the output length field value passed as next parameter.
- For columns with the edited length of less than 255, the address of a variable area (that is, the target area) with a maximum of 254 bytes where the exit must place the result of the action.
- The length of the exit result must be within the edited length passed as the next parameter. If the length of the exit's result is less than the output length, it must be padded with blanks.

EDITED OUTPUT

During initialization calls:

The address of output table.

There is a 1-to-1 correspondence between the entries of this table and SQLVAR entries in the SQLDA.

You can modify the default edited length of the columns (described in Chapter 6, "Programming and Operation Considerations") with the length less than 254. Note that the modified edited length cannot be more than the default edited length. The edited length of variable string columns with length greater than 254 must be left intact. VISION:Inquiry uses this length to format the output line. Following are the output table entries and their lengths:

Column name length	Column name	Edited length	Reserved
2	30	2	8

Note that the column name will be "." when there is a string of zero length in the SQLNAME of SQLDA.

During column calls:

The address of a halfword containing the maximum length value of the output result for the column in process (edited length). The exit routine can not modify this value during column call stage.

SEGMENT NAME

During initialization and column calls:

NAME Is the address of an 8-byte field containing the keyword "EXECSQL". By checking this field, the exit can determine whether it is a native SQL syntax inquiry or not.

TABLE NAME

During initialization and column calls:

Is the address of the area which contains the SQL SELECT statement used in the inquiry. The first two bytes of this area contain the length of the SELECT statement which follows it.

Conversion Exit IXXUCON

IXXUCON, the conversion exit, performs data conversions on all fields that are specified as 'TYPE=U' or 'TYPE=0-9' in your MAPGEN. The exit is entered each time one of these fields is referenced in an inquiry.

Use

When the conversion exit is used for multiple fields that are defined in one or more maps, you should define the respective fields as unique field types 0-9. (See Chapter 4, "The Definition Process", for information about the FIELD control statement of MAPGEN.) This technique helps you identify the map in which the field is defined. Through selection logic in your exit, you can determine the action to be performed upon the particular field.

Output Length

The output length of the fields defined as 'TYPE=U, 0-9' can be different from the fields as they are stored in the database/VSAM file. For example, if a date stored in packed format is converted by the conversion exit to a printable character format with separating slashes between day, month, and year, the output field length must be large enough to reflect the new length.

This output length must be reflected in the OUTLTH parameter of the FIELD statement in MAPGEN.

DBTYPE parameter

For fields of DB2 tables and views, you also need to specify the DBTYPE parameter of the FIELD statement. This allows you to tell DB2 the data type in which you want the column returned. (See Chapter 4, "The Definition Process", for information about the OUTLTH parameter.)

Types of Processing

The conversion exit can perform the following types of processing:

- A table lookup where a database/VSAM field is used as an argument to search a table and the corresponding function is displayed.
- (For IMS (DL/I) databases) Concatenated keys where a segment key is made up of several fields with different formats. Enter one string of characters in the inquiry and the conversion routine converts each part of the key to its format in the database. The reverse can also occur when displaying the key.

- Special editing where a 6-character date field is edited to be displayed with slashes.

The module name for the conversion exit must be IXXUCON.

IXXUCON Parameters

The parameter list passed to the exit follows:

OFFSET	CONTENT
0	PSB
4	EXIT INDICATOR
8	SOURCE
12	TARGET
16	SOURCE DATA DESCRIPTOR
20	TARGET DATA DESCRIPTOR
24	SEGMENT NAME
28	DATABASE/VSAM FILE NAME
32	STACK

IXXUCON Parameter Explanations

PSB — The address of the PCB address list passed to VISION:Inquiry when the PSB was scheduled.

EXIT INDICATOR — The address of the exit indicator.

Byte 2 (Call type):

X'80'	External to internal.
X'40'	External to external.
X'20'	Internal to internal.

Byte 3 (Return code): error indicator.

The three possible types of conversion are shown in Byte 2 of the exit indicator:

- EXTERNAL TO INTERNAL (X'80')
  Refers to the conversion of character text from an inquiry that must be converted to the form maintained in the database/VSAM file.
- INTERNAL TO EXTERNAL (X'40')
  Refers to the reverse of the above. This conversion is necessary when displaying a field.
- INTERNAL TO INTERNAL (X'20')
  Refers to the conversion of data in the form that is maintained on the database/VSAM file to the internal form required by VISION:Inquiry to perform an arithmetic operation. The exit must convert the data to a packed decimal format with a scale specified in the target data descriptor. This scale (DDFLDS) is relative to 32. This means that no decimal places would have the scale 32; +1 would be 33, -1 would be 31, and so on. You must specify the length.

IXXUCON Considerations

The following considerations should be observed when calling the user exit:

- DISPLAY of a user field (TYPE=U, TYPE=0-9) results in an INTERNAL to EXTERNAL call.
- Direct assignment of a user field to a temporary field (%FUNC=userfield) does not result in any call. The temporary field is the same type as the user field.
- Use of a user field in an expression assigned to a temporary field (%FUNC=userfield*1) results in an INTERNAL to INTERNAL call.
- Use of TOTAL or AVERAGE with a user field results in an INTERNAL to INTERNAL call.
- Use of a user field in conditional selection comparison results in a call, as shown in the following table. It should be noted that all user data type compares are character compares.

Left Side Field	:	Right Side Field	Call
User	:	Value	EXTERNAL to INTERNAL to convert value to user type once during decoding
User	:	Character	EXTERNAL to INTERNAL to convert right side to user type each time comparison is done
User	:	Numeric	None; always false
User	:	User	None, even if left and right sides are different user types
Character	:	User	INTERNAL to EXTERNAL to convert right side to character
Numeric	:	User	None; always false

Figure 7-6 User Fields in Conditional Selection Comparisons

- When an INTERNAL to INTERNAL call is made, the user conversion exit must return a packed field with length and scale as specified in the target data descriptor.
- Specification of a user field as a key or search field, or comparison of a user field with a key or search field, may cause unpredictable results.
- When an EXTERNAL to INTERNAL type of conversion occurs, VISION:Inquiry calls the user conversion exit routine before finishing the decoding process. As a result, information such as database or VSAM file name and segment name is incompletely stored and is not ready to use for this exit routine.

SOURCE — The address of the source data field. The contents of the source data field are:

- EXT to INT — A 1-byte field that contains the length minus one of the external data. This is followed by the external data. For example, if the data contains a date entered as 08/31/45, the source field would be X'07F0F861F3F161F4F5'.
- INT to EXT and INT to INT — The data field exactly as it appears in the database/VSAM file.

TARGET — The address of a variable area with a maximum length of 255 bytes where the exit must place the result of the conversion. The first byte of the target area is a length field in which the exit must place the length minus one of the result. This is followed by the result. The result must be as follows:

- **EXT to INT** — The result must be in the format that it appears in the database/VSAM file.
- **INT to EXT** — The result must be in the format that is to appear when displayed.
- **INT to INT** — The result must be in packed format with the scale specified in the target date descriptor.

SOURCE DATA DESCRIPTOR — The address of the data descriptor for the source user data field involved in conversion.

TARGET DATA DESCRIPTOR — The address of the data descriptor for the target data field. This address is only used for internal to internal conversions.

SEGMENT NAME — The content of the segment name depends on the type of file to which the exit is referring.

- For VSAM non-hierarchical data sets, it is the address of the 3-byte field that contains the character "," followed by the type of VSAM file:

  ",KS" for KSDS
  ",ES" for ESDS
  ",RR" for RRDS

- For IMS (DL/I) databases and VSAM hierarchical data sets, the content of the segment name is the address of an 8-byte segment name, the name of the most recently retrieved segment, not necessarily the segment containing the field being converted.
- For DB2 tables, the content of the segment name is the address of an 8-byte field containing the word "DB2SEG".

DATABASE or VSAM FILE NAME — The address of the 8-byte database or VSAM file name.

STACK — The address of the base address for the VISION:Inquiry stack. The stack contains the internal representation of the inquiry statement.

If the exit detects an error during conversion, it must set Byte 3 of the exit indicator to X'80' before returning. Then VISION:Inquiry provides a conversion error message to the terminal just as it does for normal conversions.

User Function Exit IXXUFNC

IXXUFNC, the function exit, provides additional processing capabilities to VISION:Inquiry. Through this exit one or more object fields or literals can be processed in order to produce a result.

Use

There is only one function exit. It is entered when a function name is specified in your inquiry and all of the specified database/VSAM fields are present. It is subsequently entered for databases when any of the fields change in any of the involved segments.

In addition to specifying the function name, you must specify the result length and scale of the values to be returned by the exit routine. These values must be specified in the inquiry statement in the exact order the exit expects them. The exit is passed addresses for the values specified on the function statement.

Since only one function exit is available in VISION:Inquiry, you must include selection logic within your exit to process additional functions. This logic can determine the appropriate action to take depending upon the function name.

Types of Processing

A technique for processing multiple functions through one exit is described later in this chapter.

The function exit can perform many different types of processing such as:

- Conversions for hexadecimal displays
- Bit testing

The module name for the function exit must be IXXUFNC.

When specifying a user function in the DISPLAY statement, the user function must be the last item specified in the statement.

IXXUFNC Parameters

The parameter list passed to the exit follows:

OFFSET	CONTENT
0	PSB
4	EXIT INDICATOR
8	STACK
12	FUNCTION DESCRIPTOR

IXXUFNC Parameter Explanations

PSB	The address of the PCB address list passed to VISION:Inquiry when the PSB was scheduled.
EXIT INDICATOR	The address of the exit indicator word.
STACK	The address of the base address for the VISION:Inquiry stack. The stack contains the internal representation of the inquiry statement.
FUNCTION DESCRIPTOR	The address of the function descriptor. (The function descriptor fields are discussed later in this chapter.)

Invoking IXXUFNC, the Function Exit

The function exit is invoked by the function statement.

The format of the function statement is:

%name(length scale)=USER(parm$_1$ parm$_2$ ... parm$_n$)

The function statement parameters are:

%name	This is a temporary field as well as the function name. This name is also used as the result field for the function.
length	The length of the field for output. If it is not specified, 8 is assumed.
scale	The scale of the field for output. If it is not specified, 0 is assumed.
parm$_1$ through parm$_n$	The field name constants and literals that are to be passed to the function exit.

FDVALUE - Function Value Array

In Assembler written exits, the variables passed to the function are accessed through the function value array (FDVALUE), which is defined in the IXUSER macro. This array contains various fields containing information pertinent to function processing. (See the IXUSER macro described in Assembler Exit Considerations on page 7-7.)

- The address of the array is calculated by adding the function value array offset (FDVALO) to the stack address. This is the address of the first variable length entry in the array.
- The address of the next element is calculated by adding five plus the value length (FDVLTH) to the address of the current element.
- The total number of elements in the array is contained in the field FDVALN.

By adding the data descriptor offset of an element in the array to the stack address, it is possible to obtain the address of the descriptor for the field should this become necessary. However, the offset is zero for the first element (the result) and all the literals that are passed.

The invocation of the function exit that follows explains this further.

Sample of Invoking the Function Exit

Assume that the database field named SALARY is a 5-byte packed field containing two decimal places. The function invoked calculates an employee's monthly salary.

The function exit is invoked by the following function statement:

```
%MO.SAL=USER(SALARY 12)
```

Note: For FDFLDS, the scale factor is relative to 32, that is, no decimal places would have a scale of 32; +1 would be 33, -1 would be 31, and so on.

Function Descriptor

When the function exit is invoked, the function descriptor contains the following:

FDFLDT	C	(1-byte character field)
FDFLDL	7	(1-byte numeric field, default of 8 minus 1)
FDFLDS	32	(1-byte numeric field, default of 32)
FDVALN	3	(1-byte numeric field, number of operands)
FDNAMEL	5	(2-byte numeric field, default of 6 minus 1)
FDNAME	MO.SAL	(32-byte character field)

Value Array

The value array contains the following:

FDVOFF	data descriptor offset
FDVFLAG	flag
FDVLTH	length-1 of the value
FDVVAL	value of field

Function Value Array

The function value array contains the following in hexadecimal:

	(FDVOFF) Database Offset	(FDVLAG) Flag	(FDVLTH) LTH-1	(FDVVAL) Value
Element 1 (Result)	0000	40	07	4B 4B 4B 4B 4B 4B 4B 4B
Element 2	mn (mn is the offset of the field SALARY.)	40	04	00 08 50 00 0C
Element 3 (Literal)	0000	40	1	F1 F2

Figure 7-7 The Function Value Array in Hexadecimal

After processing by the function exit is complete, the results of the value in elements 2 and 3 are placed in element 1. The exit sets Byte 3 of the exit indicator to X'80' to indicate the result is to be used. The exit then returns to VISION:Inquiry.

Security Exit IXXUSEC

IXXUSEC, the security exit, provides security restrictions not based on the standard mechanisms, transaction code and terminal name. In the system database, the TRANCODE/TERM points to a user directory that contains the maps and vocabulary available.

With the security exit, a user written routine can determine the user directory based on other than the standard factors. This directory is used to translate and control the inquiry.

The security exit is called by VISION:Inquiry under two circumstances:

- Prior to translating the inquiry in order to determine which directory is used.
- In cases where a message switch is used to route output as a result of the OUTPUT command. The user exit may validate if the destination terminal may receive output from this user.

The security exit is also called by AQF prior to displaying the Database/File Selection panel in order to determine which directory is used.

The module name for this exit must be IXXUSEC.

IXXUSEC Parameters

The parameter list passed to the exit follows:

OFFSET	CONTENTS
0	EIB
4	EXIT INDICATOR
8	TRANSACTION
12	TERM
16	USER RETURN ARGUMENT 1
20	USER RETURN ARGUMENT 2

IXXUSEC Parameter Explanations

EIB	The address of the CICS Exec Interface Block
EXIT INDICATOR	The address of the exit indicator. Byte 2 (Call type): X'00' exit is entered at translation or called by AQF. X'10' exit is entered at message switch. Byte 3 (Return code): Set one of the following values in the return code field of the exit indicator. ■ When called at translation (Byte 2=X'00'): The user routine should set Byte 3 of the exit indicator to X'00' to have VISION:Inquiry use the directory associated with the TRANCODE/TERM in the system database. This is the normal way VISION:Inquiry processes. Byte 3 should be set to X'80' to indicate that VISION:Inquiry is to use the user return arguments to search for the directory. Byte 3 should be set to X'40' to force termination of the inquiry. ■ When called at message switch (Byte 2=X'10'): The user routine should set Byte 3 to X'00' to indicate that VISION:Inquiry is to validate the output TERM name against the TERMs associated with the current directory. This is the normal way VISION:Inquiry processes. Byte 3 should be set to X'80' to indicate that VISION:Inquiry is to validate the TERM against all TERMs associated with all directories. This permits routing output to any terminal defined in the system database. Byte 3 should be set to X'40' to force termination of the inquiry.
TRANSACTION	The address of the 8-byte transaction name.
TERM	The address of the 8-byte terminal name.

USER RETURN ARGUMENT 1	The address of an 8-byte area where the exit routine is to place the APPL name (transaction identifier) to be used to search the system database, when Byte 3 of the exit indicator is set to X'80'.
USER RETURN ARGUMENT 2	The address of an 8-byte area where the exit routine is to place the directory name to be used to search the system database when Byte 3 of the exit indicator is set to X'80'.

VSAM Exit IXXUVSM

IXXUVSM, the VSAM exit, expands the VSAM variable records read by VISION:Inquiry to match with its MAPGEN definition.

- The length specified in the record statement of the MAPGEN should be equal to or greater than the length of the expanded record.
- VISION:Inquiry calls this exit once for each user VSAM file record read.
- This exit is commonly used for variable VSAM records in which the existence of selected fields can be checked through the flags in the record.
- This exit is used when a VSAM data set contains records with a fixed part and one or more trailers. These trailers exit based on flags in the fixed part of the record.
  In this case, the MAPGEN for the file is defined to the system with all the fields of the fixed part and the trailers.

Through this exit, the flags for each record can be checked and the record will be expanded with dummy fields for the trailers which do not exist. Thus, the record layout matches its MAPGEN definition.

The module name for this exit must be IXXUVSM.

IXXUVSM Parameters

The parameter list passed to the exit is as follows:

OFFSET	LENGTH	CONTENT
0	4	I/O AREA
4	4	RECORD LENGTH
8	4	FILE NAME

IXXUVSM Parameter Explanations

I/O AREA	The address of the I/O area which contains the VSAM record read.
RECORD LENGTH	The address of a 2-byte area which contains the length of the record read.
FILE NAME	The address of an 8-byte area which contains the ddname of the VSAM file.

If the exit changes the length of the record, the new record length should be placed in the 2-byte record length area passed to the exit.

Extract Exit IXCUEXT

IXCUEXT, the extract exit, assigns different destination IDs to different users of the EXTRACT command in the CICS online environment. VISION:Inquiry passes the destination ID (DYL1) to the exit routine before processing the EXTRACT command. The exit routine can replace the destination ID with any new destination ID (for example by checking the user ID) and return it back to VISION:Inquiry to continue processing.

IXCUEXT Parameters

The parameter list passed to the exit is:

OFFSET	LENGTH	CONTENT
0	4	DEST.ID

IXCUEXT Parameter Explanation:

DEST.ID Specifies the address of the 4-byte area that contains the extra partition transient data set destination ID for the EXTRACT command.

Assembler Extract Exit Example

Figure 7-8 illustrates the basic skeleton of the Assembler extract exit. This exit changes the supplied destination ID (DYL1) to the terminal ID.

```
*ASM             XOPTS(NOEPILOG,CICS)
DFHEISTG         DSECT
                 DFHEISTG
IXCUEXT          IXBEGIN TYPE=USER,CICS=YES
*
*                THIS IS A SAMPLE SKELETON CICS EXTRACT EXIT WHICH
*                ASSIGNS THE TERMINAL ID AS THE DESTINATION ID FOR
*                EXTRACT COMMAND
*
                 USING PARMLIST,R4                      ADDRESSABILITY FOR PARMLIST
*
*
*                REG2  IS THE BASE REGISTER
*                REG8  IS THE CICS DYNAMIC AREA, DFHEISTG
*                REG10 IS THE CICS EIB ADDRESS
*                REG12 MUST NOT BE ALTERED
*
*
                 EXEC     CICS ADDRESS EIB(R10)
                 L        R5,ADCTID                        GET THE ADDRESS OF DCT. ID
                 MVC      0(4,R5),EIBTRMID                 CHANGE DCT. ID AS TERM. ID
*
                 DFHEIRET
*
                 LTORG
*
*
PARMLIST         DSECT
ADCTID           DS A
*
                 END
```

Figure 7-8 Example Skeleton of an Assembler Extract Exit

Sort Exit IXCUSRT

IXCUSRT, the sort exit, deletes the output records at the end of online sort processing, to improve the response time. This exit can only be used when the VSAM work data set technique is used for online sort processing. The exit routine can start a new transaction which deletes the records with a generic key starting with the CICS four character terminal id from the VSAM work data set.

Two sample routines for the sort exit are provided in the II.TCVLSRC library, members IXCUSRTS and IXCUSDL, and discussed in Appendix B, “Sample User Exits”.

CICS Output Handling Exit IXCUOUT

IXCUOUT, the CICS output handling exit, handles the output data in a way other than the standard VISION:Inquiry output processing which uses the BMS paging facility. The output report will be passed to this exit line by line and the exit has the full control of handling the output. The module name for this output must be IXCUOUT.

- This exit is not used with the VISION:Journey facility.
- It is only used with the Intraccess option when sending the output report to another terminal or printer.

Invoking the Exit

Whether the inquiry contains the OUTPUT command or not, the exit will be invoked differently.

With no OUTPUT command, the exit will be started:

- At the beginning of each page of output to send the heading map.
- To output each line of data.

With the OUTPUT command, the exit will be started in different instances of sending the output to the originating and destination terminals.

IXCUOUT Parameters

The parameter list passed to the exit is:

OFFSET	CONTENT
0	Exit indicator
4	Output line
8	Work area

IXCUOUT Parameter Explanations

EXIT INDICATOR

The address of a 2- byte exit indicator.

Byte 1 (Output process stage):

- X'80' At the top of page, ready to send the heading map.
- X'40' At the detail line, ready to send the output data line.
- X'20' There is no more output line to process. This is the last invocation of the exit.
- X'08' The OUTPUT command is in process and ready to send the RTEFLIP map.
- X'04' The OUTPUT command is in process and ready to send the first heading map to the destination terminal.
- X'02' The OUTPUT command is in process and the output processing for the destination terminal is completed.

Byte 2 (Return code):

- X'00' The exit has not processed the output. VISION:Inquiry will send the output to the terminal using the BMS paging facility.
- X'80' The output has been processed by the exit. VISION:Inquiry will bypass the output processing.

OUTPUT LINE

The address of the output data line. The address is dependent on the content of byte one in the exit indicator:

X'80', X'20', X'08', X'04', or X'02' in byte one, the address is zero.

X'40' in byte one, the address is pointing to the 256 bytes output area which contains the actual output data line.

WORK AREA

Address of the work area. The work area is a 4-byte area, contains zero in the first invocation of the exit, and remains intact by VISION:Inquiry in the subsequent invocations of the exit. It can be used to pass information in different invocations of the exit.

Notes:

- If the output is handled by the exit (X'80' in byte two of the exit indicator), the process of the footing map should also be handled by the exit routine.
- The number of data lines on each screen is calculated as follows: "screen height" minus "heading map height" minus "footing map height" where heading map height is the number of lines sent by the heading map to the screen (at X'80' in byte one of the exit indicator) and footing map height is the maximum number of lines reserved for footing (the default is 2 and can be changed in the CICS parameters module).
- The process of OUTPUT command is as follows:
  - The heading map is sent to the originating terminal.
  - A message is sent to the originating terminal indicating the output will be sent to the destination terminal.
  - The RTEFLIP map is sent to purge the existing page(s) from the originating terminal.
  - The CICS ROUTE command is issued to route the output to the destination terminal.
  - If any error is encountered in the process of ROUTE command, the heading map followed by an error message will be sent to the originating terminal.
  - If no error exists, the output process on the destination terminal starts by sending the heading map followed by the data lines.
  - When the output is completed on the destination terminal, the last page of output consisting of the heading map followed by the end of inquiry message will be sent to the originating terminal.

A System Vocabulary and Codes

The following tables show the VISION:Inquiry standard vocabulary. Noise words, verbs, qualifiers, operators, and name combinations are shown with function codes. Selected words have synonyms and symbols.

NOISE WORDS

Noise word	Synonym	Code	Noise word	Synonym	Code
A		'0100'	IS		'0100'
AN		'0100'	NO		'0100'
ARE		'0100'	NONE		'0100'
AS		'0100'	OF		'0100'
BY		'0100'	ON		'0100'
FROM		'0100'	THAN		'0100'
HAD		'0100'	THE		'0100'
HAS		'0100'	TO		'0100'
HAVE		'0100'	WAS		'0100'
IN		'0100'	WERE		'0100'

VERBS

Verb	Synonym	Code	Verb	Synonym	Code
AVERAGE		'1600'	EXTRACT	E	'1C00'
CONTINUE		'1B00'	FIND	I, INTER	'1100'
COUNT		'1400'	LIMIT		'1700'
DEFER		'1A00'	OUTPUT		'1000'
DEFINE		'1800'	SORT		'1200'

VERBS					
Verb	**Synonym**	**Code**	**Verb**	**Synonym**	**Code**
DELETE		‘1900’	SUM		‘5C00’
DISPLAY	D, PRINT	‘1300’	TOTAL		‘1500’
EDITSQ		‘1C31’			

UDO					
Verb	**Synonym**	**Code**	**Verb**	**Synonym**	**Code**
COLUMN	COL	‘4006’	PF		‘4009’
DATE		‘4001’	REPEAT	PF	‘4007’
EDIT		‘4008’	SHOW		‘1E00’
FORMAT		‘0008’	SKIP		‘4004’
LINE		‘4003’	SPACE	SP	‘4005’
NOSPACE	NOSP	‘400A’	TIME		‘4002’
PAGE		‘4000’	DATEF		‘4008’

VISION:Journey			RESERVED WORDS (NOT MODIFIABLE)	
Verb	**Synonym**	**Code**	**Non-SQL**	**SQL = non-SQL plus:**
PCEXTRACT	PCE	‘1C01’	OPTIONS	EXECSQL
PCDELETE	PCD	‘1C11’	INCLUDE	ENDEXEC
PCLOAD	PCL	‘1C21’	EXCLUDE	

QUALIFIERS, SYNONYMS AND CODES					
Qualifier	**Synonym**	**Code**	**Qualifier**	**Synonym**	**Code**
ASCENDING	ASC	‘5600’	INQUIRY		‘0003’
COMMENT		‘000A’	LTERM	TERM	‘0004’
DATA		‘0006’	MAP		‘0002’

QUALIFIERS, SYNONYMS AND CODES					
Qualifier	**Synonym**	**Code**	**Qualifier**	**Synonym**	**Code**
DEFERRED		‘0030’	SYSTEM		‘0020’
DESCENDING	DSC	‘5680’	USER		‘5E00’
DESCRIPT		‘0009’	VOCABULARY	VOCAB	‘0005’
DIRECTORY	DIRECT	‘0010’	WHOLE		‘000F’
FUNCTION		‘0007’			

- For Ascending and Descending, use only the synonyms.

OPERATORS, SYNONYMS, SYMBOLS AND CODES							
Operator	**Syn**	**Sym**	**Code**	**Operator**	**Syn**	**Sym**	**Code**
EQUAL	EQ	=	‘5000’	(LEFT PARENTHESIS)		(	‘6600’
LT		<	‘5004’	(RIGHT PARENTHESIS)		)	‘6700’
GT		>	‘5008’	(SEMICOLON)		;	‘6C00’
LE		<=	‘5010’	ALL			‘3080’
GE		>=	‘5014’	IF	WITH		‘3000’
NE		¬=	‘5018’	FIRST			‘5100’
LIKE			‘501C’				
(ADDITION)		+	‘5800’	LAST			‘5200’
(SUBTRACTION)		-	‘5900’	AND		&	‘6000’
(MULTIPLICATION)		*	‘5A00’	OR		|	‘6100’
(DIVISION)		/	‘5B00’	ESCAPE			‘5F00’

- For Addition, Subtraction, Multiplication, Division, Parentheses and Semicolon, use only the symbols.

NAME COMBINATIONS, SYNONYMS AND CODES		
Name combination	**Synonym**	**Code**
DEFER INQUIRY	DI	‘1A03’
CONTINUE DEFERRED INQUIRY	CDI	‘1B33’
DEFINE DIRECTORY INQUIRY	DDI	‘1813’
DEFINE DIRECTORY FUNCTION	DDF	‘1817’
DISPLAY DIRECTORY DATA	PDD	‘1316’
DISPLAY DIRECTORY DATA WHOLE	PDD WHOLE	‘1325’
DISPLAY DIRECTORY DESCRIPT	PDDDS	‘131F’
DISPLAY DIRECTORY FUNCTION	PDF	‘1317’
DISPLAY DIRECTORY INQUIRY	PDI	‘1313’
DISPLAY DIRECTORY INQUIRY WHOLE	PDIW	‘1322’
DISPLAY DIRECTORY INQUIRY COMMENT	PDIC	‘131D’
DISPLAY DIRECTORY INQUIRY COMMENT WHOLE	PDICW	‘132C’
DISPLAY DIRECTORY LTERM	PDL, PDT	‘1314’
DISPLAY DIRECTORY MAP	PDM	‘1312’
DISPLAY DIRECTORY MAP WHOLE	PDMW	‘1321’
DISPLAY DIRECTORY TERM	PDL, PDT	‘1314’
DISPLAY DIRECTORY VOCABULARY	PDV	‘1315’
DISPLAY FORMAT	FD	‘1308’
DISPLAY SYSTEM	PS	‘1320’

B Sample User Exits

Sample User Exits

The installation source library, II.TCVLSRC, contains exit samples in Assembler and in PL/I. Exit names and exit sample names are:

Exit Name	Exit Sample Name	Associated Routine	Description
IXXUCON	IXXUCONP IXXUCONS		Conversion exit PL/I Conversion exit Assembler
		IXHEX	Macro to convert fields to hexadecimal
IXXUFNC	IXXUFNCS		Interface to user function exits Function exit (single function only)
		IXXUTFTM	Bit testing function
IXXUIN	IXXUINS		Input exit
		IXXSTABL	Input exit security table
IXXUOUT	IXXUOUTS		Output exit
IXXUSEC	IXXUSECS		Security exit
IXCUSRT	IXCUSRTS		Sort exit
		IXCUSDL	Sample delete routine associated with transaction IQDV

In the following sections, you will find sample exits and considerations for using them. Samples can be implemented in their entirety or used as guides to designing and coding your own conversion exits with your specific requirements in mind.

Sample Conversion Exits (IXXUCONP and IXXUCONS)

The purpose of the sample conversion exits (IXXUCONP and IXXUCONS) illustrated in this appendix is to provide VISION:Inquiry with a hexadecimal capability. With the IXXUCON exit, fields defined to VISION:Inquiry can be displayed in hexadecimal format, or data can be entered into the fields in hexadecimal format.

The IXXUCON exit can be necessary for the following reasons:

- When the exact contents of a field need to be displayed (even if the field is invalid or non-displayable). This is particularly useful when debugging an application or looking at a bad segment in the database.
- When the fields involved in a selection criteria do not have a character representation (binary fields) or a special bit configuration.

IXXUCONP PL/I Sample

Source library member II.TCVLSRC (IXXUCONP) contains the following sample user conversion exit written in PL/I. IXXUCONP demonstrates hexadecimal capability.

```
IXXUCON:  PROC (I1,I2,I3,I4,I5,I6,I7,I8,I9);
  DCL  (I1,I2,I3,I4,I5,I6,I7,I8,I9)    FIXED BIN (15,0);
  DCL  (P1,P2,P3,P4,P5,P6,P7,P8,P9)    PTR;
  DCL  CONV_LEN                        FIXED BIN (15,0);
  DCL  1  BINARY_LENGTH     BASED(ADDR(CONV_LEN)),
          2  FILL                      BIT (8) ALIGNED,
          2  BIN_LEN                   BIT (8) ALIGNED;
  DCL  PSB_LIST (15) BASED (P1)        PTR;
  DCL  1  EXIT_INDICATOR BASED (P2),
          2  BYTE (4)                  BIT (8) ALIGNED;
  DCL  1  SOURCE BASED (P3),
          2  SOURCE_LTH                BIT (8) ALIGNED,
          2  SOURCE_DATA               CHAR (255);
  DCL     SOURCE_CHAR CHAR(256) BASED (P3);
  DCL  1  TARGET BASED (P4),
          2  TARGET_LTH                BIT (8) ALIGNED,
          2  TARGET_DATA               CHAR (255);
  DCL  DB_NAME       BASED (P8)        CHAR (8);
  DCL  1  SOURCE_DATA_DESC BASED (P5),
          2  SDD_FUNC        FIXED BIN (15,0),
          2  SDD_LTH         FIXED BIN (15,0),
          2  SDD_SEG         FIXED BIN (15,0),
          2  SDD_FLDO        FIXED BIN (15,0),
          2  SDD_FLDT        CHAR (1),
          2  SDD_FLDE        BIT (8) ALIGNED,
          2  SDD_FLDL        BIT (8) ALIGNED,
          2  SDD_FEDL        BIT (8) ALIGNED,
          2  SDD_FLDS        BIT (8) ALIGNED,
          2  SDD_SUBO        BIT (8) ALIGNED,
          2  SDD_SUBL        BIT (8) ALIGNED,
          2  SDD_MAPID       CHAR (1),
          2  SDD_DDNAMEL     FIXED BIN (15,0),
          2  SDD_NAME        CHAR (32),
          2  SDD_IMSNME      CHAR (8);
  DCL   (IX1,IX2)            FIXED BIN (15,0);
  DCL   IP_FIELD (254)       BIT (4);
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 1 of 4)

```
  DCL   IP_WORK              CHAR (127)  BASED(ADDR(IP_FIELD));
  DCL   OP_FIELD (254)       CHAR (1);
  DCL   OP_WORK              CHAR (254)  DEFINED OP_FIELD;
  DCL   1 CONVERT_INPUT,
          2  BIT0            BIT (4)  INIT('0000'B) ALIGNED,
          2  CHAR0           CHAR (1) INIT('0'),
          2  BIT1            BIT (4)  INIT('0001'B) ALIGNED,
          2  CHAR1           CHAR (1) INIT('1'),
          2  BIT2            BIT (4)  INIT('0010'B) ALIGNED,
          2  CHAR2           CHAR (1) INIT('2'),
          2  BIT3            BIT (4)  INIT('0011'B) ALIGNED,
          2  CHAR3           CHAR (1) INIT('3'),
          2  BIT4            BIT (4)  INIT('0100'B) ALIGNED,
          2  CHAR4           CHAR (1) INIT('4'),
          2  BIT5            BIT (4)  INIT('0101'B) ALIGNED,
          2  CHAR5           CHAR (1) INIT('5'),
          2  BIT6            BIT (4)  INIT('0110'B) ALIGNED,
          2  CHAR6           CHAR (1) INIT('6'),
          2  BIT7            BIT (4)  INIT('0111'B) ALIGNED,
          2  CHAR7           CHAR (1) INIT('7'),
          2  BIT8            BIT (4)  INIT('1000'B) ALIGNED,
          2  CHAR8           CHAR (1) INIT('8'),
          2  BIT9            BIT (4)  INIT('1001'B) ALIGNED,
          2  CHAR9           CHAR (1) INIT('9'),
          2  BIT10           BIT (4)  INIT('1010'B) ALIGNED,
          2  CHAR10          CHAR (1) INIT('A'),
          2  BIT11           BIT (4)  INIT('1011'B) ALIGNED,
          2  CHAR11          CHAR (1) INIT('B'),
          2  BIT12           BIT (4)  INIT('1100'B) ALIGNED,
          2  CHAR12          CHAR (1) INIT('C'),
          2  BIT13           BIT (4)  INIT('1101'B) ALIGNED,
          2  CHAR13          CHAR (1) INIT('D'),
          2  BIT14           BIT (4)  INIT('1110'B) ALIGNED,
          2  CHAR14          CHAR (1) INIT('E'),
          2  BIT15           BIT (4)  INIT('1111'B) ALIGNED,
          2  CHAR15          CHAR (1) INIT('F');
     DCL  1  CONVERT_TABLE      BASED(ADDR(CONVERT_INPUT)),
          2  CNVRT_TBL (16),
             3  BIT_DATA     BIT (4) ALIGNED,
             3  CHAR_DATA    CHAR (1);
     P1 = ADDR (I1);
     P2 = ADDR (I2);
     P3 = ADDR (I3);
     P4 = ADDR (I4);
     P5 = ADDR (I5);
     P8 = ADDR (I8);
/*============================================================*/
/*     TEST FOR CONVERSION EXIT INDICATOR                     */
/*     IF NOT CONVERSION EXIT, RETURN                         */
/*============================================================*/
   IF EXIT_INDICATOR.BYTE (1) ¬= '00100000'B THEN RETURN;
/*============================================================*/
/*    RETURN IF INTERNAL TO INTERNAL CONVERSION               */
/*============================================================*/
   IF EXIT_INDICATOR.BYTE (2) = '00100000'B THEN RETURN;
/*============================================================*/
/*    IF NOT FIELD TYPE U INDICATE CONVERSION ERROR           */
/*    AND RETURN                                              */
/*============================================================*/
     IF SDD_FLDT ¬= 'U' THEN
     DO;
        EXIT_INDICATOR.BYTE (3) = '10000000'B;
        RETURN;
     END;
/*============================================================*/
/*    CHECK FOR INTERNAL TO EXTERNAL CONVERSION               */
/*       CHARACTER TO HEX CONVERSION                          */
/*============================================================*/
   IF EXIT_INDICATOR.BYTE (2) = '01000000'B THEN
      DO;
         CALL INT_EXT_CONV;
         RETURN;
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 2 of 4)

```
      END;
/*============================================================*/
/*    CHECK FOR EXTERNAL TO INTERNAL CONVERSION               */
/*          HEX TO CHARACTER CONVERSION                       */
/*============================================================*/
   IF EXIT_INDICATOR.BYTE (2) = '10000000'B THEN
      DO;
         CALL EXT_INT_CONV;
         RETURN;
      END;
/*============================================================*/
/*       INTERNAL TO EXTERNAL CONVERSION ROUTINE              */
/*       CHARACTER TO HEX CONVERSION                          */
/*============================================================*/
INT_EXT_CONV:  PROC;
/*============================================================*/
/*     TEST FOR DATABASE FILE NAME                            */
/*     IF THE DATABASE FILE NAME IS NOT IIDBDDM RETURN        */
/*============================================================*/
   IF DB_NAME  ¬= 'IIDBDDM ' THEN RETURN;
/*============================================================*/
/*       RETURN WITH CONVERSION ERROR IF FIELD > 127 BYTES    */
/*============================================================*/
   IF BIN(SDD_FLDL,15) > 126 THEN
   DO;
      EXIT_INDICATOR.BYTE (3) = '10000000'B;
      RETURN;
   END;
   CONV_LEN = (BIN(SDD_FLDL,15) + 1) * 2;
   IP_WORK = SOURCE_CHAR;
   DO IX2 = 1 TO CONV_LEN;
      DO IX1 = 1 TO 16;
        IF IP_FIELD(IX2) = BIT_DATA(IX1) THEN
        DO;
           OP_FIELD(IX2) = CHAR_DATA(IX1);
           GOTO CONV_FOUND;
        END;
      END;
      /*============================================================*/
      /* CONVERSION ERROR  -  BIT STRING NOT IN TABLE             */
      /*============================================================*/
      EXIT_INDICATOR.BYTE (3) = '10000000'B;    /*  X'80'  */
      RETURN;
CONV_FOUND:
   END;
   CONV_LEN = CONV_LEN - 1;
   TARGET_LTH = BIN_LEN;
   TARGET_DATA = SUBSTR(OP_WORK,1,CONV_LEN+1);
END INT_EXT_CONV;
/*============================================================*/
/*       EXTERNAL TO INTERNAL CONVERSION ROUTINE              */
/*       HEX TO CHARACTER CONVERSION                          */
/*============================================================*/
EXT_INT_CONV:  PROC;
/*============================================================*/
/*       RETURN WITH CONVERSION ERROR IF FIELD > 254 BYTES    */
/*============================================================*/
   IF BIN(SOURCE_LTH,15) > 253 THEN
   DO;
      EXIT_INDICATOR.BYTE (3) = '10000000'B;
      RETURN;
   END;
   CONV_LEN = BIN(SOURCE_LTH,15) + 1;
   OP_WORK = SOURCE_DATA;
   DO IX2 = 1 TO CONV_LEN;
      DO IX1 = 1 TO 16;
        IF OP_FIELD(IX2) = CHAR_DATA(IX1) THEN
        DO;
           IP_FIELD(IX2) = BIT_DATA(IX1);
           GOTO CHAR_FOUND;
        END;
      END;
      /*============================================================*/
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 3 of 4)

```
      /* CONVERSION ERROR  -  CHARACTER NOT IN TABLE          */
      /*=====================================================*/
      EXIT_INDICATOR.BYTE(3) = '10000000'B;   /* X'80'  */
      RETURN;
CHAR_FOUND:
   END;
   CONV_LEN = (CONV_LEN / 2) - 1;
   TARGET_LTH = BIN_LEN;
   TARGET_DATA = SUBSTR(IP_WORK,1,CONV_LEN+1);
END EXT_INT_CONV;
END IXXUCON;
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 4 of 4)

IXXUCONS Assembler Sample

Source library member II.TCVLSRC (IXXUCONS) contains the source code for the sample input conversion exit written in Assembler.

```
         TITLE 'SAMPLE USER CONVERSION EXIT'
*
*        THIS SAMPLE USER EXIT CONVERTS THE EMPLOYEE NUMBER ON
*        THE PLANT DATABASE TO A BIRTH DATE IN THE FORM MM/DD/YY.
*        THE USER FIELD THAT CAUSES THIS ROUTINE TO BE CALLED IS
*        'EMP.DOB'. THIS STANDS FOR EMPLOYEE DATE OF BIRTH. ANY
*        TIME THE FIELD 'EMP.DOB' IS REFERENCED IN AN INQUIRY, THIS
*        ROUTINE WILL USE THE EMPLOYEE NUMBER AND AN INTERNAL TABLE
*        TO MAKE THE CONVERSION.
*
IXXUCON  IXBEGIN TYPE=USER
         EJECT
         USING PARMLIST,R4
*
*        CHECK EXIT INDICATOR FOR PROPER CALL
*
         L     R1,AEXIT                POINT TO EXIT INDICATOR
         TM    0(R1),X'20'             CONVERSION EXIT?
         BZ    RETURN                  RETURN IF NOT
*
*        CHECK FOR DATABASE NAME
*
         L     R1,ADBNAME              POINT TO DATABASE NAME
         CLC   0(8,R1),=C'IIDBDDM '    PLANT DATABASE?
         BNE   RETURN                  RETURN IF NOT
*
*        CHECK DATA DESCRIPTOR FOR 'EMP.DOB'
*
         L     R1,ASDD                 POINT TO SOURCE DATA DESCRIPTOR
         USING DD,R1
         LH    R11,DDNAMEL             LOAD NAME LENGTH
         CH    R11,=H'6'               IS LENGTH 7?
         BNE   RETURN                  RETURN IF NOT
         CLC   DDNAME(7),=C'EMP.DOB'   RETURN IF NOT
         BNE   RETURN                  RETURN IF NOT
         DROP  R1
*
*        DETERMINE TYPE OF CONVERSION TO BE DONE
*
         L     R1,AEXIT                POINT TO EXIT INDICATOR
         TM    1(R1),X'80'             EXT TO INT CONVERSION?
         BO    EXTINT                  BRANCH IF IT IS
         TM    1(R1),X'40'             INT TO EXT CONVERSION?
         BO    INTEXT                  BRANCH IF IT IS
*
*        AN INTERNAL TO INTERNAL CONVERSION CAN ONLY INVOLVE
```

Figure B-2 IXXUCONS Source (Assembler) - IXXUCON Conversion Exit (Page 1 of 3)

```
*        ARITHMETIC FIELDS. A CONVERSION ERROR IS INDICATED
*        AND CONTROL RETURNED TO CALLER.
*
         OI    2(R1),X'80'              INDICATE CONVERSION ERROR
         B     RETURN                   RETURN TO CALLER
*
*        EXTERNAL TO INTERNAL CONVERSION
*
EXTINT   EQU   *
         L     R5,ASOURCE               POINT TO SOURCE
         LA    R6,DOBTABL               POINT TO TABLE
         CLI   0(R5),X'07'              LENGTH OF EXTERNAL DATA 8?
         BE    EXTINT05                 CONTINUE IF IT IS
         OI    2(R1),X'80'              INDICATE CONVERSION ERROR
         B     RETURN                   RETURN TO CALLER
EXTINT05 EQU   *
         CLC   0(5,R6),=C'99999'        END OF TABLE?
         BE    EXTINT10                 USE EMP.NO 99999 IF IT IS
         CLC   5(8,R6),1(R5)            COMPARE TABLE TO SOURCE
         BE    EXTINT10                 FINISHED IF EQUAL
         LA    R6,13(,R6)               POINT TO NEXT ENTRY
         B     EXTINT05                 CONTINUE SEARCH
EXTINT10 EQU   *
         L     R7,ATARGET               POINT TO TARGET
         MVI   0(R7),X'04'              INDICATE LENGTH OF 5
         MVC   1(5,R7),0(R6)            MOVE EMP.NO TO TARGET
         B     RETURN                   RETURN TO CALLER
*
*        INTERNAL TO EXTERNAL CONVERSION
*
INTEXT   EQU   *
         L     R5,ASOURCE               POINT TO SOURCE
         LA    R6,DOBTABL               POINT TO TABLE
INTEXT05 EQU   *
         CLC   0(5,R6),=C'99999'        END OF TABLE?
         BE    INTEXT10                 USE EMP.DOB 99/99/99 IF IT IS
         CLC   0(5,R6),0(R5)            COMPARE TABLE TO SOURCE
         BE    INTEXT10
         LA    R6,13(,R6)               POINT TO NEXT ENTRY
         B     INTEXT05                 CONTINUE SEARCH
INTEXT10 EQU   *
         L     R7,ATARGET               POINT TO TARGET
         MVI   0(R7),X'07'              INDICATE LENGTH OF 8
         MVC   1(8,R7),5(R6)            MOVE EMP.DOB TO TARGET
         B     RETURN                   RETURN TO CALLER
*
RETURN   EQU   *
         IXRETURN
         EJECT
DOBTABL  EQU   *
         DC    C'1005008/31/45'
         DC    C'1010004/16/23'
         DC    C'2032706/27/50'
         DC    C'2085001/31/16'
         DC    C'2090005/27/39'
         DC    C'3017509/15/41'
         DC    C'3042512/25/98'
         DC    C'3045010/20/25'
         DC    C'3050007/04/41'
         DC    C'3050202/29/49'
         DC    C'3062505/03/56'
         DC    C'4025006/30/54'
         DC    C'4045003/09/46'
         DC    C'5012504/31/44'
         DC    C'5062008/15/34'
         DC    C'5072506/06/32'
         DC    C'6032707/22/57'
         DC    C'6085011/30/89'
         DC    C'9999999/99/99'
*
         LTORG
         EJECT
```

Figure B-2 IXXUCONS Source (Assembler) - IXXUCON Conversion Exit (Page 2 of 3)

```
PARMLIST DSECT
APSB     DS    A
AEXIT    DS    A
ASOURCE  DS    A
ATARGET  DS    A
ASDD     DS    A
ATDD     DS    A
ASGNAME  DS    A
ADBNAME  DS    A
ASTKADR  DS    A
*
         IXUSER
         END
```

Figure B-2 IXXUCONS Source (Assembler) - IXXUCON Conversion Exit (Page 3 of 3)

Adding the IXHEX Conversion Macro to IXXUCON

The IXHEX macro converts fields to hexadecimal format and needs to be coded within the IXXUCON control section (CSECT).

The macro expects the addressability to the VISION:Inquiry parameters to be established prior to its invocation. This can be accomplished through the IXBEGIN macro and an Assembler USING statement for the PARMLIST, which defines the address list.

OPERATION	OPERAND
IXHEX	**PREFIX=HEX**

Where:

PREFIX = HEX The prefix for the field names, defined in the MAPGEN, to be processed by the hexadecimal routines. All the 'TYPE=U' field names containing the prefix are processed by the routines.

If this parameter is not specified, it defaults to 'HEX'.

When defining the prefix, care must be taken that the prefix plus the name lengths do not violate the maximum length for the field names specified in the APPL statement NAMELENGTH parameter.

Figure B-3 contains the source code for the IXHEX macro. This macro should be added to your appropriate installation macro library.

```
             MACRO
             IXHEX     &PREFIX=HEX
*
             LCLA &A
             AIF (K'&PREFIX LE 31).OK1
             MNOTE     12,'PREFIX LENGTH GREATER THAN 31. MACRO IGNORED'
             MEXIT
*
.OK1         AIF ('&PREFIX'(1,1) GE 'A' AND '&PREFIX'(1,1) LE 'Z').OK2
                MNOTE   12,'PREFIX MUST BEGIN WITH ALPHABETIC CHARACTER. MACRO IGNORED'
             MEXIT
*
.OK2         ANOP
&A           SETA      K'&PREFIX
             EJECT
*
* MACRO      IXHEX -                                          - V.1.00
             SPACE
             DC        0H'0'                  ALIGN ON HALF WORD BOUNDARY
             L         R1,AEXIT               ADDRESS OF ACTION WORD
             TM        0(R1),X'20'            CONVERSION EXIT?
             BZ        IXHEXEND               NO, BYPASS
*
             L         R3,ASDD                ADDRESS OF SOURCE DESCRIPTOR
             USING     DD,R3                  ADDRESSABILITY FOR DSECT
             CLC       DDNAME(&A),=C'&PREFIX' IS FIELD TO BE PROCESSED?
             BNE       IXHEXEND               NO, BYPASS
*
             L         R5,ASOURCE             ADDRESS OF SOURCE DATA
             L         R7,ATARGET             ADDRESS OF TARGET DATA
*
             TM        1(R1),X'40'            INTERNAL TO EXTERNAL?
             BO        IXHEXINT               YES, PROCESS
             TM        1(R1),X'80'            EXTERNAL TO INTERNAL?
             BO        IXHEXEXT               YES, PROCESS
             B         IXHEXEND               BYPASS
*
IXHEXINT     DS        0H                     INTERNAL TO EXTERNAL CONVERSION
             BAL       R11,INTEXTHX           LINK TO ROUTINE
             B         IXHEXEND               RETURN TO CALLER
*
IXHEXEXT     DS        0H                     EXTERNAL TO INTERNAL CONVERSION
             BAL       R11,EXTINTHX           LINK TO ROUTINE
             B         IXHEXEND               RETURN TO CALLER
             SPACE     5
INTEXTHX     DS        0H
             SR        R9,R9                  CLEAR REG 9
             IC        R9,DDFLDL              FIELD LENGTH - 1
             LA        R9,1(R9)               ACTUAL FIELD LENGTH
             CH        R9,=H'127'             IS LENGTH GREATER THAN 127?
             BH        0(R11)                 YES, RETURN TO CALLER
             SLL       R9,1                   MULTIPLY BY 2 TO GET EXTERNAL LENGTH
             LR        R6,R9                  SAVE REG 9
             BCTR      R6,0                   OUTPUT FIELD LENGTH - 1
             STC       R6,0(R7)               STORE LENGTH IN TARGET FIELD
             LA        R7,1(R7)               POINT PAST LTH INDICATOR IN TARGET
*
LOOPHEX1     DS        0H
             CH        R9,=H'14'              LENGTH GREATER THAN 14?
             BL        ELOOPHX1               NO, SKIP
             UNPK      0(15,R7),0(8,R5)       UNPACK 8 BYTES
             TR        0(15,R7),IXTABLE.      TRANSLATE
             SH        R9,=H'14'              SUBTRACT 14 FROM EXTERNAL LENGTH
             LTR       R9,R9                  ALL DONE?
             BZ        0(R11)                 YES, RETURN TO CALLER
             LA        R7,14(R7)              ADJUST TARGET POINTER
             LA        R5,7(R5)               ADJUST SOURCE POINTER
             B         LOOPHEX1               LOOP BACK
*
```

Figure B-3 IXHEX Macro Source (Page 1 of 2)

```
ELOOPHX1    DS      0H
            LR      R6,R9                   SAVE REMAINDER (TARGET)
            SRL     R6,1                    DIVIDE BY 2 TO GET INTERNAL LENGTH
            SLL     R9,4                    SHIFT FOUR BITS LEFT
            OR      R9,R6                   CONCATENATE BOTH LENGTHS
            STC     R9,UNPKINST+1           STORE IN INSTRUCTION
            SRL     R9,4                    RESTORE REMAINDER
            STC     R9,TRINST1+1            STORE IN INSTRUCTION
UNPKINST    UNPK    0(0,R7),0(0,R5)         UNPACK N BYTES
TRINST1     TR      0(0,R7),IXTABLE         TRANSLATE N BYTES
            BR      R11                     RETURN TO CALLER
            SPACE   5
EXTINTHX    DS      0H
            SR      R6,R6                   CLEAR REG 6
            IC      R6,0(R5)                GET EXTERNAL LENGTH - 1
            LA      R6,1(R6)                COMPUTE ACTUAL LENGTH
            LR      R10,R6                  SAVE REG 6
            SRL     R10,1                   DIVIDE BY 2 TO GET INTERNAL LENGTH
            BCTR    R10,0                   COMPUTE INTERNAL LENGTH - 1
            STC     R10,0(R7)               STORE IT IN TARGET FIELD
*
LOOPHEX2    DS      0H
            CH      R6,=H'14'               IS EXTERNAL LENGTH GE 14?
            BL      ELOOPHX2                NO, BYPASS
            TR      1(14,R5),IXTABLE        TRANSLATE NON-NUMERIC DATA
            PACK    1(8,R7),1(15,R5)        PACK TO STRIP OFF ZONE
            SH      R6,=H'14'               SUBTRACT 14 FROM EXTERNAL LENGTH
            LTR      R6,R6                  ALL DONE?
            BZ       0(R11)                 YES, RETURN TO CALLER
            LA       R5,7(R5)               ADJUST SOURCE POINTER
            LA       R7,14(R7)              ADJUST TARGET POINTER
            B        LOOPHEX2               LOOP BACK
*
ELOOPHX2    DS       0H
            STC      R6,TRINST2+1           STORE LENGTH IN INSTRUCTION
            LR       R10,R6                 SAVE REG 6
            SRL      R10,1                  DIVIDE BY 2 TO GET INTERNAL LENGTH
            SLL      R10,4                  SHIFT LEFT FOUR BITS
            OR       R10,R6                 AND IN EXTERNAL LENGTH
            STC      R10,PACKINST+1         STORE IN INSTRUCTION
TRINST2     TR       1(0,R5),IXTABLE        TRANSLATE NON-NUMERIC DATA
PACKINST    PACK     1(0,R7),1(0,R5)        PACK TO STRIP OFF ZONE
            BR       R11                    ALL DONE, RETURN TO CALLER
            SPACE    5
IXTABLE     DC       256X'00'               TRANSLATE TABLE
            ORG      IXTABLE+C'0'
            DC       C'0123456789ABCDEF'
            ORG
            ORG      IXTABLE+C'A'
            DC       X'0A0B0C0D0E0F'
            ORG
            SPACE    5
            LTORG
            SPACE    5
            DS       0H                      ALIGN ON A HALF WORD BOUNDARY
IXHEXEND    EQU      *
            DROP     R5
            EJECT
            MEXIT
            MEND
```

Figure B-3 IXHEX Macro Source (Page 2 of 2)

IXXUCON Conversion Exit Considerations

The fields to be processed by the IXXUCON conversion exit must be previously defined to VISION:Inquiry by a FIELD statement in the MAPGEN and must adhere to the following rules.

- The TYPE parameters must be coded as `U'.
- The OUTLTH parameters must be coded as being twice the value specified in the LENGTH parameters.
- All the field names must contain a common prefix. This prefix is defined when installing the exit and uniquely identifies the `TYPE = U' fields to be processed by the hexadecimal routine in the conversion exit.

Figure B-4 illustrates FIELD statements used with the sample conversion exit.

```
SEGMENT SEGM=PLANT,PARENT=0,BYTES=40,KEY=PLANTKEY
 FIELD START=1,LENGTH=5,TYPE=C,NAME=PLANT.ID,KEY=SEQ-U
 FIELD START=1,LENGTH=5,TYPE=U,NAME=HEX.PLANT.ID,KEY=SEQ-U,
     OUTLTH=10,KEYPOS=1
 FIELD START=6,LENGTH=2,TYPE=C,NAME=PLANT.REGION
 FIELD START=8,LENGTH=7,TYPE=C,NAME=PLANT.PHONE
 FIELD START=15,LENGTH=25,TYPE=C,NAME=PLANT.NAME
*
SEGMENT SEGM=PROD,PARENT=PLANT,BYTES=35,KEY=PRODKEY
 FIELD START=1,LENGTH=2,TYPE=C,NAME=PROD.CODE,KEY=SEQ-U
 FIELD START=3,LENGTH=4,TYPE=B,NAME=PROD.QTY
 FIELD START=7,LENGTH=4,TYPE=B,NAME=PROD.AMT,SCALE=+2
 FIELD START=7,LENGTH=4,TYPE=U,NAME=HEX.PROD.AMT,
     OUTLTH=8
 FIELD START=11,LENGTH=25,TYPE=C,NAME=PROD.DESC
*
SEGMENT SEGM=EMP,PARENT=PLANT,BYTES=31,KEY=EMPKEY
 FIELD START=1,LENGTH=5,TYPE=N,NAME=EMP.NO,KEY=SEQ-U,
     OUTEDIT=NONE
 FIELD START=6,LENGTH=1,TYPE=C,NAME=EMP.SEX
 FIELD START=7,LENGTH=25,TYPE=C,NAME=EMP.NAME
 FIELD START=7,LENGTH=25,TYPE=U,NAME=HEX.EMP.NAME,
     OUTLTH=50
*
SEGMENT SEGM=SAL,PARENT=EMP,BYTES=11,KEY=SALKEY
 FIELD START=1,LENGTH=2,TYPE=N,NAME=SAL.YEAR,KEY=SEQ-U
 FIELD START=3,LENGTH=5,TYPE=P,NAME=SAL.YTD,SCALE=+2
 FIELD START=3,LENGTH=5,TYPE=U,NAME=HEX.SAL.YTD,
     OUTLTH=10
 FIELD START=8,LENGTH=4,TYPE=P,NAME=SAL.DED,SCALE=+2
 FIELD START=8,LENGTH=4,TYPE=Y,SUBSTR=(1,5),NAME=SAL.DED.INT
 FIELD START=8,LENGTH=4,TYPE=Y,SUBSTR=(6,2),NAME=SAL.DED.DEC
*
```

Figure B-4 FIELD Statements Using the Sample Conversion Exit

Invoking the IXXUCON Conversion Exit

Fields processed by the IXXUCON conversion exit are specified in inquiries the same as any other fields defined in the MAPGEN.

Figure B-5 illustrates specifications of inquiries containing fields processed by the sample conversion exit.

```
INQUIRY1:  DISPLAY PLANT PLANT.ID HEX.PLANT.ID PROD.CODE
           PROD.QTY PROD.AMT HEX.PROD.AMT;
INQUIRY2:  DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME
           HEX.EMP.NAME SAL.YTD HEX.SAL.YTD;
INQUIRY3:  DISPLAY PLANT PLANT.NAME EMP.NO EMP.NAME
           IF HEX.PLANT.ID = 'F1F0F1F0F0';
```

Figure B-5 Inquiries Containing Fields to be Processed by the Conversion Exit

Figure B-6, Figure B-7, and Figure B-8 illustrate the output generated from the inquiries in Figure B-5, as a result of the conversion exit processing.

PLANT.ID	HEX.PLANT.ID	PROD.CODE	PROD.QTY	PROD.AMT	HEX.PROD.AMT
10100	F1F0F1F0F0				
20150	F2F0F1F5F0	PO	30	29.95	00000BB3
		RO	450	18.95	00000767
		SC	300	38.00	00000ED8
		SJ	78	58.00	000016A8
		SR	94	28.95	00000B4F
		TR	180	45.00	00001194
30200	F3F0F2F0F0				
40300	F4F0F3F0F0	AS	48	98.00	00002648
		CD	105	79.00	00001EDC
		HW	76	99.00	000026AC
		TS	34	65.00	00001964
50300	F5F0F3F0F0	CC	88	65 00	00001964
		DD	78	60.00	00001770
		EM	110	75.00	00001D4C
		MA	307	15.95	0000063B
		RH	39	45.00	00001194
		SS	135	90.00	00002328
60200	F6F0F2F0F0	DA	14,970	3.95	0000018B
		DC	20,550	4.95	000001EF
		DF	11,289	4.50	000001C2
		DH	381	55.00	0000157C
		DO	1,300	15.95	0000063B
		KS	12,340	7.95	0000031B
		PB	789	3.95	0000018B
		PG	126	29.95	00000BB3
		TB	470	18.95	00000767
70500	F7F0F5F0F0	CR	88	65.00	00001964
		DA	14,970	3.95	0000018B
		DC	20,550	4.95	000001EF
		DD	78	60.00	00001770
		DH	331	55.00	0000157C
		DO	1,300	15.95	0000063B
		EM	110	75.00	00001D4C
		KS	12,340	7.95	0000031B
		MA	307	15.95	0000063B
		PB	789	3.95	0000018B
		PG	126	29.95	00000BB3
		PO	30	29.95	00000BB3
		RH	39	45.00	00001194

Figure B-6 Output Generated from the Conversion Exit in INQUIRY1 (Page 1 of 2)

```
                                    RO                  450         18.95     00000767
                                    SC                  300         38.00     00000ED8
                                    SJ                   78         58.00     000016A8
                                    SR                   94         28.95     00000B4F
                                    SS                  135         90.00     00002328
                                    TB                  470         18.95     00000767
                                    TR                  180         45.00     00001194
IXX9121 END OF INQUIRY                    (60,7 USER DB CALLS, ROOTS)
```

Figure B-6 Output Generated from the Conversion Exit in INQUIRY1 (Page 2 of 2)

```
PLANT.ID EMP NO EMP NAME           HEX.EMP NAME                                        SAL YTD  HEX.SAL.YTD

10100    10103  WILLIAM AMES       E6C903D3C9C1D440C1D4C5E240404B4B404040404B4B4B4040  52,000.00  005200000C
                                                                                       64,000.00  006400000C
         10104  PHYLLIS LOCKMEYER  D7CBEB03D3C9E240D3D6C3D2D4C5EBC5D940404040404040  48,000.00  004800000C
                                                                                       59,000.00  005900000C
         10105  MARY ANN THOMAS    04C1D9EB40CID5D540E3C8D6D4CIE24040404B404B40404040  15,600.00  001560000C
20150    21116  WILMA FORD         E6C9D3D4C140C6D6D9C44040404040404040404040404040  15,600.00  001560000C
                                                                                       18,800.00  001880000C
         21124  CHARLES SALTER     C3C8C1D9D3C3E240E2C103E3C5D9404040404040404040404040  24,000.00  002400000C
                                                                                       30,000.00  003000000C
                                                                                       39,000.00  003900000C
         21137  PETER ZATKIN       D7C5E3C5D940E9C1E3D2C9D5404040404040404040404040  44,000.00  004400000C
                                                                                       50,000.00  005000000C
                                                                                       56,000.00  005600000C
         21164  SUSAN WARE         E2E4EZC1D540E6C1D9C54040404040404B4B40404040404040  32,000.00  003200000C
                                                                                       41,000.00  004100000C
30200    30201  JOHN HENRY CRANE   D1D6C8D540C8C5D5D9EB40C3D9C1D5C5404040404040404040  13,400.00  001340000C
                                                                                       19,600.00  001960000C
                                                                                       22,000.00  002200000C
         30202  FREDERICH GRAY     C6D9C5C4C5D9C9C3C840C7D9C1E8404B404040404040404040  48,000.00  004800000C
                                                                                       59,000.00  005900000C
         30205  MITCHELL J HOOPS   D4C9E3C3C8C5D3D340D140C8D6D6D7E2404B40404040404040  76,000.00  007600000C
                                                                                       92,000.00  009200000C
                                                                                       98,000.00  009800000C
         30207  JANE LOWELL        D1C1D5C540D3D6E6C5D3D3404040404840404040404040404040  22,000.00  002200000C
                                                                                       26,000.00  002600000C
         30211  PATRICIA BLAKELY   D7C1E3D9C9C3C9C140C2D3C1D2C5D3EB404040404040404040  30,000.00  003000000C
         30215  SHARON DALEY       E2CBC1D9D6D940C4C1D3C5EB4040404040404040404040404040  17,000.00  001700000C
40300    40304  DONALD M KING      C4D6DSC1D3C440D440D2C905C7404040404040404040404040  48,000.00  004800000C
                                                                                       66,000.00  006600000C
                                                                                       75,000.00  007500000C
         40306  JOAN EVANS         D1D6C1DS40C5E5C1D5E2404040404040404040404040404040  64,000.00  006400000C
                                                                                       73,200.00  007320000C
50300    50304  JONATHAN OAKS      D1D6D5C1E3C8C1D540D6C1D2E2404040404040404040404040  36,000.00  003600000C
                                                                                       46,000.00  004600000C
         50322  MADELYN BATES      D4C1C4CSD3EBD340C2C1E3C5E2404040404040404040404040  32,000.00  003200000C
         50323  VICKY WARD         ESC9C3D2EB40E6C1D9C44040404040404040404040404040  20,000.00  002000000C
60200    60205  DAVID YORK         C4C1E3C9C440EBD4D9D2404040404040404040404040404040  31,000.00  003100000C
                                                                                       37,000.00  003700000C
                                                                                       43,000.00  004300000C
         60209  RUSSELL M SIMMONS  D9E4E2E2C5D3D3400440E2C9D4D4D6D5E24040404040404040  48,000.00  004800000C
                                                                                       60,000.00  006000000C
         60251  MARCIE MORINO      D4C1D9C3C9CS40D4D6D9C9DS0640404040404040404040404040  31,000.00  003100000C
         60258  KAREN REDFERN      D2C1D9C5D540D9CSC4C6C3D9D5404040404040404040404040  22,000.00  002200000C
                                                                                       26,000.00  002600000C
7050     70511  RONALD T JACKSON     D9D6D5C1D3C440E340D1C1C3D3E2D6D5404040404040404040  32,000.00  003200000C
                                                                                       39,000.00  003900000C
                                                                                       46,000.00  004600000C
                                                                                       52,000.00  005200000C
         70519  STEPHEN MCGEE        E2E3C5D7CBC5D540D4CAC7C5C5404040404040404040404040404B  19,000.00  001900000C
                                                                                       23,000.00  002300000C
         7052   AGNES COVINGTON      C1C7D5C5E240C3D6E5C9D5C7E3D6D5404040404040404040404040  20,000.00  002000000C
                                                                                       23,000.00  002300000C
         70529  MARTHA WALLINGHAM    D4C1D9E3CBC140E6C1D3D3C9D5C7C8C1D44040404040404040  33,000.00  003300000C
IXX9121 END OF INQUIRY           (60,7 USER DB CALLS, ROOTS)
```

Figure B-7 Output Generated from the Conversion Exit in INQUIRY2

```
PLANT.NAME     EMP.NO         EMP.NAME

DALLAS SALES   10103          WILLIAM AMES
               10104          PHYLLIS LOCKMEYER
               10105          MARY ANN THOMAS



IXX9121 END OF INQUIRY              (5,1 USER DB CALLS, ROOTS)
```

Figure B-8 Output Generated from the Conversion Exit in INQUIRY3

Implementing the Conversion Exit

A sample conversion exit, such as IXXUCONP or IXXUCONS, must be compiled or assembled. The resultant object module must then be included in an SMP/E USERMOD and applied to the VISION:Inquiry load modules. See Chapter 7, "User Exits" for an example of a USERMOD for a user exit and the sample SMP/E JCL.

Sample Function Exit (IXXUFNC)

In the VISION:Inquiry system, there is only one user function exit, regardless of the number of actual functions being used in the installation. This also means that all the functions are coded as one program, which is not always desirable or practical for implementation and maintenance purposes.

The IXXUFNC was written to provide an easy way to implement user function exits. Although referred to as "the function exit" or "the generic function exit," IXXUFNC technically is not an exit. IXXUFNC is an interface between the VISION:Inquiry system and the user function exits.

- IXXUFNC takes care of the specific linkage conventions, and based on the function name, passes control to the specific user exit, defined in a function exit user table. Control goes back to VISION:Inquiry upon return from the exit.
- Each user function exit can now be coded as a separate module, with standard linkage conventions and can be totally independent of all the other user exits.
- Each user function exit can be either resident in the VISION:Inquiry nucleus or dynamically loaded at execution time, which allows for modifying the exit without having to re-link the whole VISION:Inquiry nucleus every time (for test purposes).

- The same function can be invoked several times in the same inquiry without causing a name conflict.
- To add a new function exit to the system, you only need to modify a table (the function name table) that contains the function name and corresponding address. The new standalone exit can then be incorporated in the system.

The exits described on the following pages can be used in their entirety, or as an example for designing and coding your own exits.

Source Code for the Sample IXXUFNC Function Exit

Figure B-9 contains the source code for the sample IXXUFNC function exit.

```
                PRINT NOGEN,DATA
*ASM            XOPTS(NOEPILOG,CICS)
IXXF            TITLE '*** USER FUNCTION ROUTINE: GENERIC - IXXUFNC ***'
DFHEISTG        DSECT
                DFHEISTG
IXXUFNC         IXBEGIN TYPE=USER,CICS=YES
                SPACE 5
                USING FD,R1
                USING TABLEFD,R5
                USING FDVALUE,R11
                USING PARMLIST,R4
*
* CHECK EXIT INDICATOR FOR PROPER CALL
*
                SPACE
                L       R1,AEXIT                POINT TO EXIT INDICATOR
                TM      0(R1),X'10'             IS IT A FUNCTION EXIT?
                BZ      RETURN                  NO, RETURN
                SPACE 3
*
* CHECK FUNCTION NAME
*
                SPACE
                L       R1,AFD                  POINT TO FUNCTION DESCRIPTOR
                LA      R5,FDTABLE              ADDRESS OF ALLOWABLEFUNCTIONS
                LA      R6,NUMELEM              NUMBER OF ELEMENTS IN TABLE
LOOPTABL        DS      0H
                L       R3,FDTLTH               FUNCTION NAME LENGTH - 1
                EX      R3,CLCFDNAM             COMPARE NAMES
                BE      FDFOUND                 STOP LOOKING IF EQUAL
                LA      R5,LFDTABLE(R5)         TRY NEXT ENTRY
                BCT     R6,LOOPTABL             LOOP BACK
                SPACE 3
*
* FUNCTION NAME NOT IN TABLE
*
                SPACE
                LA      R6,=CL8'INVFUNC ' INDICATE ERROR
ERRORTNE        DS      0H
                L       R11,ASTACK              ADDRESS OF STACK
                AH      R11,FDVALO              ADD OFFSET TO VALUE ARRAY
                SR      R7,R7                   CLEAR REG 7
                IC      R7,FDVLTH               GET FIELD LENGTH
                CH      R7,=H'7'                IS LENGTH GREATER THAN 7
                BNH     LTHLESS                 NO, SKIP
                LA      R7,7                    FORCE MAXIMUM
LTHLESS         DS      0H
                EX      R7,MVCERR               MOVE ERROR MESSAGE
                L       R1,AEXIT                ADDRESS OF EXIT INDICATOR WORD
```

Figure B-9 Source Code for the IXXUFNC Exit (Page 1 of 3)

```
                OI      2(R1),X'80'             INDICATE READY
                B       RETURN                  RETURN
*
* FUNCTION NAME IN TABLE
*
                SPACE
FDFOUND         DS      0H
                L       R14,FDTMOD              ADDRESS OF MODULE
                LTR     R14,R14                 IS MODULE LINK EDITED WITH NUCLEUS?
                BZ      MODLKED                 YES, SKIP
*
                EXEC    CICS LOAD PROGRAM (TABLEFD+32) SET(R0)
*
                LR      R15,R0                  ADDRESS OF ENTRY POINT
*

                CALL    (15)                    BRANCH TO MODULE
*
                EXEC    CICS RELEASE PROGRAM (TABLEFD+32)
*
                B       RETURN                  RETURN TO CALLER
                SPACE 3
MODLKED         DS      0H
                L       R15,FDTADDR             ADDRESS OF FUNCTION ROUTINE
                LTR     R15,R15                 HAS ADDRESS BEEN RESOLVED?
                BNZ     MODULEOK                YES, PROCEED
                LA      R6,=CL8'NOMODULE'       INDICATE ERROR
                B       ERRORTNE                BRANCH TO ERROR ROUTINE
MODULEOK        DS      0H
                BALR    R14,R15                 BRANCH TO MODULE
                B       RETURN                  RETURN
                SPACE 3
*
* RETURN TO CALLER
*
                SPACE
RETURN          DS      0H
                DFHEIRET RCREG=I5               RETURN TO CALLER
                SPACE 3
*
* REMOTE INSTRUCTIONS
*
                SPACE
CLCFDNAM        CLC     FDNAME(0),FDTNAME
MVCERR          MVC     FDVVAL(0),0(R6)
                SPACE 5
                LTORG
                EJECT
*****************************************************************
*                                                               *
* FUNCTION TABLE                                                *
*                                                               *
*****************************************************************
                SPACE 5
FDTABLE         DS      0F
                SPACE
                DC      CL32'HEXA',V(IXXUFHEX),A(0),A(4-1)
LFDTABLE        EQU     *-FDTABLE
                DC      CL32'UPDX',V(IXXUFUPX),A(0),A(4-1)
                DC      CL32'UPD',V(IXXUFUPD),A(0),A(3-1)
                DC      CL32'TM',V(IXXUFTM),A(0),A(2-1)
                DC      CL32' ',CL8' ',A(0)
                SPACE 5
FDTABLED        EQU     *
NUMELEM         EQU     (FDTABLED-FDTABLE)/LFDTABLE
                EJECT
PARMLIST        DSECT
APSB            DS      A
AEXIT           DS      A
ASTACK          DS      A
AFD             DS      A
                SPACE 3
```

Figure B-9 Source Code for the IXXUFNC Exit (Page 2 of 3)

```
TABLEFD          DSECT
FDTNAME          DS       CL32
FDTADDR          DS       A
FDTMOD           DS       A
FDTLTH           DS       A
                 SPACE 3
                 IXUSER
                 SPACE 3
                 END
```

Figure B-9 Source Code for the IXXUFNC Exit (Page 3 of 3)

Function Exit Table

The function exit table contains the name and address for each executable function. It also specifies whether the module is resident or dynamically loaded.

Each function is defined by one Assembler statement, coded immediately after the 'FDTABLE' label, in the IXXUFNC module.

The statement has two different coding formats. The first statement defines a resident user function exit.

DC CL32'FNAME',V(CSECT),A(0),A(N-1)

The second statement defines a dynamically loaded user function exit.

DC CL32'FNAME',CL8'LOADNAME',A(N-1)

The parameters are:

FNAME	The function name (or the first n letters of the function name).
CSECT	The name of the first or only control section of the user function exit which will be invoked.
N	The number of characters from the function name that is used when looking up the table. It allows the same function to be used several times in the same inquiry by specifying function names with a suffix appended to FNAME.
LOADNAME	The name of the exit load module.

Note: It is good practice to define a dummy entry in the table (third statement in Figure B-10). A new function can be added by superzapping in the function name. This avoids recompiling the IXXUFNC module.

The function exit table, as it looks in Assembler, is illustrated in Figure B-10.

```
*********************************************
*                                           *
*              FUNCTION TABLE               *
*                                           *
*********************************************
FDTABLE  DS  0F
         DC  CL32'HEXA',V(IXXUFHEX),A(0),A(4-1)
         DC  CL32'TM',CL8'IXXUFTM',A(2-1)
         DC  CL32' ',CL8' ',A(0)
```

Figure B-10 The Function Table in Assembler

Whenever possible, function exit table entries should be ordered by frequency usage. The most frequently used entry should be the first one defined in the table.

Sample User Function Error Messages

The following error messages can be issued in the field defined by the function name in the inquiry:

INVFUNC	The corresponding function name has not been defined in the function exit table.
NOMODULE	The user function exit is defined as resident, but is not link edited with the VISION:Inquiry nucleus.

Installing the Sample Function Exit

To install the sample function exit:

1. Assemble the IXXUFNC function exit source.
2. Define the VISION:Inquiry macro library to the Assembler compiler as a SYSLIB data set.
3. Assemble all the resident user function exits defined in the function table. Include the object modules of the IXXFNC function exit routine and the resident user function exits into an SMP/E USERMOD and apply them to the VISION:Inquiry load module. See Chapter 7, "User Exits" for an example of a USERMOD for a user exit and the sample SMP/E JCL.
4. Assemble and catalog all the dynamically loaded user function exits defined in the function table in a library accessible during execution time (defined as STEPLIB or JOBLIB).

IXXUFTM Sample Bit Testing Function

The IXXUFTM user function provides a dynamic way of testing the bit configuration of any field defined to VISION:Inquiry. The test is performed, one byte at a time, by invoking the function.

The MAPGEN definition for the field being tested need not be modified.

To invoke the user function, specify the byte and field to be tested against an 8-character mask containing 0s or 1s. (Each character represents the corresponding bit of the byte being tested.)

The result of the test is returned as a literal in the result field, which may contain one of the following values:

ONES	When there is an exact correspondence between the bits in the field and the 1s in the mask.
ZEROES	When there is no correspondence between the bits in the field and the 1s in the mask.
MIXED	When there is some correspondence between the bits in the field and the 1s in the mask.

The result field can then be used in a conditional IF clause.

The IXXUFTM user function is implemented as a stand-alone module, called by the IXXUFNC generic function exit. Therefore, it can be invoked several times in the same inquiry without causing any name conflict.

IXXUFTM Bit Testing Function Source

Figure B-11 contains the source code for the IXXUFTM bit testing function.

```
            PRINT  NOGEN,DATA
TM          TITLE '*** USER FUNCTION ROUTINE:  TM ***'
IXXUFTM     CSECT
R0          EQU   0
R1          EQU   1
R2          EQU   2
R3          EQU   3
R4          EQU   4
R5          EQU   5
R6          EQU   6
R7          EQU   7
R8          EQU   8
R9          EQU   9
R10         EQU   10
R11         EQU   11
R12         EQU   12
R13         EQU   13
R14         EQU   14
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 1 of 4)

```
R15          EQU   15
RA           EQU   10
RB           EQU   11
RC           EQU   12
RD           EQU   13
RE           EQU   14
RF           EQU   15
             SPACE 3
             USING FD,R1                    FUNCTION DESCRIPTOR
             USING FDVALUE,R5               FUNCTION PARAMETER
             USING PARMLIST,R4              PARMLIST
             USING IXXUFTM,R10              BASE REGISTER
             SPACE 3
*
* INITIAL HOUSEKEEPING
*
             SPACE
             SAVE  (14,12),,*               SAVE REGISTERS
*
             LR    R10,R15                  ADDRESSABILITY FORCSECT
             ST    R13,TMDSAVEA+4           STORE PREVIOUS SAVEAREA ADDRESS
             LR    R3,R13                   SAVE REG13
             LA    R13,TMDSAVEA             ADDRESS OF NEW SAVEAREA
             ST    R13,8(R3)                STORE IT IN PREVIOUS SAVEAREA
             SPACE 3
*
* STORE ADDRESS OF EACH PARAMETER
*
             SPACE
             L     R1,AFD                   FUNCTION DESCRIPTOR ADDRESS
             L     R5,ASTACK                STACK ADDRESS
             AH    R5,FDVALO                ADD OFFSET
             SR    R3,R3                    CLEAR REG3
             IC    R3,FDVALN                NUMBER OF PARAMETERS PASSED
             SR    R9,R9                    CLEAR REG9
             LA    R6,ADDRTAB               ADDRESS OF TABLE
 LOOPELEM    DS    0H
             ST    R5,0(R6)                 STORE PARAMETER ADDRESS
             LA    R6,4(R6)                 BUMP TO NEXT ENTRY IN TABLE
             IC    R9,FDVLTH                LENGTH - 1 OF ELEMENT
             LA    R5,5(R9,R5)              BUMP TO NEXT ELEMENT
             BCT   R3,LOOPELEM              LOOK BACK
             SPACE 3
*
* VALIDATE NUMBER OF PARAMETERS
*
             SPACE
             CLI    FDVALN,4                 4 PARAMETERS PASSED?
             BNE    TMERROR1                 NO, ERROR
             SPACE
*
* VALIDATE PARAMETERS
*
             SPACE
             L      R5,LITERADR              ADDRESS OF LITERAL INFO
             CLI    FDVLTH,7                 IS,LENGTH 8?
             BNE    TMERROR5                 NO, ERROR
             LA     R6,FDVVAL                ADDRESS OF LITERAL
             LA     R8,BITTABLE              ADDRESS OF TABLE
             LA     R11,8                    NUMBER OF ITERATIONS
             SR     R9,R9                    CLEAR REG9
BITLOOP      DS     0H
             CLI    0(R6),C'0'               IS BYTE ZERO?
             BE     BITLOOP1                 YES,SKIP
             CLI    0(R6),C'1'               IS BYTE ONE?
             BNE    TMERROR6                 NO, ERROR
             O      R9,0(R8)                 AND IN BIT CONFIGURATION
BITLOOP1     DS     0H
             LA     R6,1(R6)                 GET NEXT BYTE IN LITERAL
             LA     R8,4,(R8)                GET NEXT ENTRY IN TABLE
             BCT    R11,BITLOOP              LOOP BACK
*
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 2 of 4)

```
            L      R5,DISPLADR             ADDRESS OF FIELD
            CLI    FDVLTH,2                IS LENGTH GT 3
            BH     TMERROR2                YES, ERROR
            IC     R3,FDVLTH               GET FIELD LENGTH
            EX     R3,TMMVN                IS FIELD NUMERIC?
            EX     R3,TMCLC                IDEM
            BNE    TMERROR3                NO, ERROR
            EX     R3,TMPACK               PACK FIELD
            CVB    R3,TMPACKD              CONVERT TO BINARY
            BCTR   R3,R0                   MINUS ONE
*
            L      R5,FIELDADR             ADDRESS OF FIELD INFORMATION
            SR     R8,R8                   CLEAR REG8
            IC     R8,FDVLTH               LENGTH OF FIELD
            CR     R3,R8                   IS DISPLACEMENT PAST FIELD
            BH     TMERROR4                YES, ERROR
            AR     R5,R3                   ADD DISPLACEMENT TO ADDRESS
            SPACE  3
*
* WE ARE NOW READY TO TEST FIELD
*
             EX    R9,TM                   TEST CORRESPONDING FIELD
             L     R5,RESULADR             ADDRESS OF RESULT FIELD
*
             BO    TMBO                    BRANCH IF ONES
             BM    TMBM                    BRANCH IF MIXED
             BZ    TMBZ                    BRANCH IF ZEROES
             MVC   FDVVAL(8),=CL8'NOT ONES' INDICATE STATUS CODE
             B     RETURN                  RETURN TO 12
*
TMBO         MVC   FDVVAL(8),=CL8'ONES'    INDICATE STATUS CODE
             B     RETURN                  RETURN TO 12
*
TMBM         MVC   FDVVAL(8),=CL8'MIXED'   INDICATE STATUS CODE
             B     RETURN                  RETURN TO 12
*
TMBZ         MVC   FDVVAL(8),=CL8'ZEROES'  INDICATE STATUS CODE
             B     RETURN                  RETURN TO 12
             SPACE
*
* FINAL HOUSEKEEPING
*
             SPACE
RETURN       DS    0H
             L     R1,AEXIT                ADDRESS OF EXIT INDICATOR
             OI     2(R1),X'80'              INDICATE READY TO USE RESULT FIELD
*
             L     R12,TMDSAVEA+4          ADDRESS OF PREVIOUS SAVEARA
*
             RETURN (14,12),RC=0           RETURN TO CALLER
             SPACE  3
*
* ERROR ROUTINES
*
             SPACE
TMERROR1     DS    0H
             L     R5,RESULADR             ADDRESS OF RESULT FIELD
             MVC   FDVVAL(8),=C'NOT4PARM' RETURN CODE
             B     RETURN
             SPACE 3
TMERROR2     DS    0H
             L     R5,RESULADR             ADDRESS OF RESULT FIELD
             MVC   FDVVAL(8),=C'WRONGDIS' RETURN CODE
             B     RETURN
             SPACE 3
TMERROR3     DS    0H
             L     R5,RESULADR             ADDRESS OF RESULT FIELD
             MVC   FDVVAL(8),=C'INVALDIS' RETURN CODE
             B     RETURN
             SPACE 3
TMERROR4     DS    0H
             L     R5,RESULADR             ADDRESS OF RESULT FIELD
             MVC   FDVVAL(8),=C'DISP2BIG' RETURN CODE
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 3 of 4)

```
              B     RETURN
              SPACE 3
TMERROR5      DS    0H
              L     R5,RESULADR              ADDRESS OF RESULT FIELD
              MVC   FDVVAL(8),=C'LITERSIZ' RETURN CODE
              B     RETURN
              SPACE 3
TMERROR6      DS    0H
              L     R5,RESULADR              ADDRESS OF RESULT FIELD
              MVC   FDVVAL(8),=C'NOT 0/1 ' RETURN CODE
              B     RETURN                   EXIT
              SPACE 3
*
* REMOTE INSTRUCTIONS
*
              SPACE
TMMVN         MVN   ZEROES(0),FDVVAL
TMCLC         CLC   ZEROES(0),FDVVAL
TMPACK        PACK  TMPACKD,FDVVAL(0)
TM            TM    FDVVAL,0
              EJECT
TMDSAVEA      DS    18F
ADDRTAB       DS    0F
RESULADR      DS    A
FIELDADR      DS    A
DISPLADR      DS    A
LITERADR      DS    A
*
TMPACKD       DS    D
ZEROS         DC    C'000'
              SPACE 5
BITTABLE      DS    0F
              DC    A(128)
              DC    A(64)
              DC    A(32)
              DC    A(16)
              DC    A(8)
              DC    A(4)
              DC    A(2)
              DC    A(1)
              SPACE 5
              LTORG
              EJECT
PARMLIST      DSECT
APSB          DS    A
AEXIT         DS    A
ASTACK        DS    A
AFD           DS    A
              SPACE 5
              IXUSER
              SPACE 5
                END
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 4 of 4)

IXXUFNC Function Exit Coding Considerations

When you implement a new function exit, code the SAVE macro instead of the IXBEGIN macro. Save area linkage should also be coded.

The registers are set as if they were initialized by the IXBEGIN macro:

R4	Contains the PARMLIST ADDRESS
R15	Contains the module entry point

Control should be returned to the IXXUFNC routine by the RETURN macro.

Invoking the IXXUFNC Function Exit

In all the following examples, the user function name is TM. The function name could very easily be changed to any other name by modifying the function exit table in module IXXUFNC.

The IXXUFTM user function follows the normal coding rules for user functions.

The format of the IXXUFTM user function is:

%TMname=USER(Fieldname n 'xxxxxxxx')

The IXXUFTM parameters are:

%TMname	The function name. By varying 'name', you can invoke the same function more than one time in the same inquiry. This also defines the result field that contains a literal, returned by the function, indicating the result of the test. VISION:Inquiry assumes a length of 8 bytes for this field.
Fieldname	The name of the field to be processed by the function. Only one byte, defined by the 'N' parameter, is tested.
n	Specifies the displacement of the byte to be tested in 'Fieldname' (starting from 1). N cannot be larger than the field length as defined to VISION:Inquiry.
'xxxxxxxx'	Represents the bit configuration to be tested. Each 'x' can either be '0' or '1'. The literal must be 8 characters long and must be enclosed in quotation marks.

Figure B-12 illustrates inquiries containing the bit testing function.

```
INQUIRY1:  PRO DISPLAY PLANT PLANT.ID HEX.PLANT.ID
           %TM.PLANTID1 = USER(PLANT.ID 1 '00000001')
           %TM.PLANTID2 = USER(PLANT.ID 1 '00000010') IF
           %TM.PLANTID1 = 'ONES' OR %TM.PLANTID2 = 'ZEROES';;
INQUIRY2:  PRO DISPLAY PLANT PLANT.ID HEX.PLANT.ID
           %TM.PLANTID1 = USER(PLANT.ID 1 '00000001')
           %TM.PLANTID2 = USER(PLANT.ID 1 '00000010') IF
           %TM.PLANTID1 = 'ONES' AND %TM.PLANTID2 = 'ZEROES';;
```

Figure B-12 Inquiries Containing the Bit Testing Function

Installing the Sample Bit Testing Function IXXUFTM

To install the sample bit testing function:

1. Assemble and link edit the Assembler language IXXUFTM function program.
2. Modify the function exit table in the IXXUFNC module to include an entry as shown below:

```
DC   CL32'TM',CL8'IXXUFTM',A(2-1)
```

3. After you modify the function exit table, re-assemble the IXXUFNC module. Include the object module of the IXXFUNC function exit routine into an SMP/E USERMOD and apply them to the VISION:Inquiry load module. See Chapter 7, "User Exits" for an example of a USERMOD for a user exit and the sample SMP/E JCL.

IXXUFTM Bit Testing Function Error Messages

The following error codes can be issued by the bit testing function. They are placed in the result field defined by the function name:

NOT4PARM	The number of parameters specified in the function is invalid.
WRONGDIS	The displacement is specified by more than 3 digits.
INVALDIS	The displacement is not numeric.
DISP2BIG	The displacement goes past the field.
LITERSIZ	The mask does not have 8 characters.
NOT 0/1	The mask is specified with characters different from 0s and 1s.

Figure B-13 illustrates an inquiry containing the bit testing function IXXUFTM.

```
INQUIRY1:  PRO DISPLAY PLANT PLANT.ID HEX.PLANT.ID
           %TM.PLANTID1 = USER(PLANT.ID 1 '00000001')
           %TM.PLANTID2 = USER(PLANT.ID 1 '00000010') IF
           %TM.PLANTID1 = 'ONES' OR %TM.PLANTID2 = 'ZEROES';;
INQUIRY2:  PRO DISPLAY PLANT PLANT.ID HEX.PLANT.ID
           %TM.PLANTID1 = USER(PLANT.ID 1 '00000001')
           %TM.PLANTID2 = USER(PLANT.ID 1 '00000010') IF
           %TM.PLNATID1 = 'ONES' AND %TM.PLANTID2 = 'ZEROES';;
```

Figure B-13 Inquiry Containing the Bit Testing Function

Figure B-14 illustrates the output of the inquiry specified in Figure B-13 after it is processed by IXXUFTM.

```
PLANT.ID     HEX.PLANT.ID     TM.PLANTID1     TM.PLANTID2

10100        F1F0F1F0F0       ONES            ZEROES
30200        F3F0F2F0F0       ONES            ONES
40300        F4F0F3F0F0       ZEROES          ZEROES
50300        F5F0F3F0F0       ONES            ZEROES
70500        F7F0F5F0F0       ONES            ONES
IXX9121 END OF INQUIRY              (8,7 USER DB CALLS, ROOTS)

10100        F1F0F1F0F0       ONES            ZEROES
50300        F1F0F1F0F0       ONES            ZEROES
IXX9121 END OF INQUIRY              (8,7 USER DB CALLS, ROOTS)
```

Figure B-14 Output Generated from the Bit Testing Function

Sample Input Exit (IXXUIN)

This sample input exit is written in Assembler. It can be implemented in its entirety, or it can be used as a guide for designing and coding your own input exit with your specific installation requirements in mind.

The purpose of the input exit illustrated in this section is to provide additional security to VISION:Inquiry.

The sample exit contains a password table that is generated by the IXSECTY and IXSGEN macros (these are discussed in Appendix C, "The IXSECTY and IXSGEN Macros"). When an inquiry is processed, the password is matched against the password table. A binary search of the table is performed. If the password does not match any of the entries, one or more error messages are placed into the inquiry, thus causing VISION:Inquiry to detect invalid statements and reject the inquiry.

Source Code for the Sample IXXUIN Input Exit

Figure B-15 contains the input exit sample source code for the IXXUIN input exit.

```
*ASM          XOPTS (NOEPILOG,CICS)
              PRINT NOGEN
UIN           TITLE '*** USER INPUT EXIT: IXXUIN - SECURITY ROUTINE ***'
DFHESTG       DSECT
              DFHEISTG
PTABADDR      DS      A
ABEGIN        DS      A
PASSWORD      DS      CL8
IXXUIN        IXBEGIN TYPE=USER,CICS=YES
              SPACE   5
RA            EQU     10
RB            EQU     11
RC            EQU     12
RD            EQU     13
RE            EQU     14
RF            EQU     15
              SPACE   5
              USING   PARMLIST,R4
              USING   PASSWINF,R7
              USING   PASSWTAB,RA
              USING   TRANTABL,R6
              USING   INPUTMSG,R3
              SPACE   3
*
* CHECK FOR TYPE OF EXIT
*
              SPACE
              L       R1,AEXIT              GET ACTION WORD ADDRESS
              TM      0(R1),X'80'           IS IT INPUT EXIT?
              BZ      RETURN                NO, RETURN
              SPACE   3
*
* FORCE DELIMITERS AT END OF TEXT
*
              L       R3,AINPUT             GET ADDRESS OF MESSAGE
              LH      R1,MSGLENTH           GET SIZE OF MESSAGE
              LTR     R1,R1                 IS LENGTH ZERO?
              BZ      RETURN                YES, RETURN
              LA      R1,4(R1,R3)           POINT TO END OF TEXT
              MVC     0(2,R1),=C';;'        MOVE DELIMITERS
              SPACE   3
*
* FIND FIRST NON-BLANK CHARACTER IN INPUT MESSAGE
*
              SPACE
              L       R3,AINPUT             GET ADDRESS OF MESSAGE
              LA      R7,MESSAGE            GET ADDRESS OF TEXT
              LA      R1,80                 SET LOOP COUNTER
LOOPNBLK      DS      0H
              CLI     0(R7),C' '            IS CHARACTER BLANK?
              BNE     NBLNKFND              NO, STOP LOOKING
              LA      R7,1(R7)              BUMP ONE BYTE
              BCT     R1,LOOPNBLK           LOOP BACK
              B       PNOTFND               PASSWORD NOT IN FIRST 80 BYTES
              SPACE   3
*
* FIND FIRST BLANK IN INPUT MESSAGE
*
              SPACE
NBLNKFND      DS      0H
              ST      R7,ABEGIN             STORE PASSWORD ADDRESS
              LA      R1,9                  SET LOOP COUNTER
LOOPBLNK      DS      0H
              CLI     0(R7),C' '            IS IT A BLANK?
              BE      BLNKFND               YES, STOP LOOKING
              LA      R7,1(R7)              BUMP ONE BYTE
```

Figure B-15 Sample Input Exit IXXUIN (Page 1 of 4)

```
             BCT     R1,LOOPBLNK            LOOP BACK
             B       PLNTHERR               PASSWORD HAS MORE THAN 8
CHARACTERS
             SPACE   3
*
* ISOLATE AND MOVE PASSWORD TO WS
*
             SPACE
BLNKFND      DS      0H
             S       R7,ABEGIN              COMPUTE NUMBER OF BYTES
             BCTR    R7,0                   MINUS ONE FOR MVC INSTRUCTION
             L       R5,ABEGIN              GET PASSWORD ADDRESS
             MVC     PASSWORD,=CL8' '       BLANK OUT PASSWORD
             EX      R7,MVCPASS             MOVE IT
             EX      R7,MVCBLNK             BLANK OUT PASSWORD
             EJECT
*
* DETERMINE IF IXXSTABL IS IN CORE AND, IF NOT, LOAD IT
*
             SPACE
             L       R7,ATABLE              LOAD VCON
             LTR     R7,R7                  IS CSECT LINK EDITED?
             BNZ     TABLEIN                YES, BYPASS
*
             EXEC    CICS LOAD PROGRAM ('IXXSTABL') SET (R0)
*
             ST      R0,ATABLE              SAVE ENTRY POINT
             OI      ATABLE,X'80'           FLAG IT AS LOADED
             LR      R7,R0                  LOAD CSECT ADDRESS
             SPACE   3
*
* SEARCH PASSWORD TABLE
*
             SPACE
TABLEIN      DS      0H
             L       R5,PASSWTAD            GET PASSWORD TABLE ADDRESS
             ST      R5,PTABADDR            SAVE ADDRESS
             LH      RF,PASSWNUM            GET NUMBER OF PASSWORDS (H)
             LA      RE,1                   INITIALIZE (L)
SEARCH       DS      0H
             CR      RF,RE                  COMPARE H / L
             BL      NOTFOUND               EXIT IF LOW
             LR      RB,RE                  SAVE L
             AR      RB,RF                  K = H + L
             SRL     RB,1                   K = (H + L) / 2
             LR      R9,RB                  SAVE K
             BCTR    R9,0                   MINUS 1 TO INDEX
             MH      R9,PASSWTLN            MULTIPLY BY LENGTH OF TABLE
ELEMENT
             L       RA,PTABADDR            GET TABLE ADDRESS
             AR      RA,R9                  ADD DISPLACEMENT
*
             CLC     PASSWORD,PASSW         COMPARE ARGUMENTS
             BE      PFOUND                 STOP LOOKING IF EQUAL
*
             BH      ARGHIGH                SKIP IF ARGUMENT HIGHER
             BCTR    RB,0                   K = K - 1
             LR      RF,RB                  H = K
             B       SEARCH                 LOOP BACK
ARGHIGH      DS      0H
             LA      RE,1(RB)               L = K + 1
             B       SEARCH                 LOOP BACK
             SPACE   3
*
* WE HAVE FOUND THE PASSWORD IN THE TABLE.  WE NOW CHECK IF TRANCODE IS
*   ALLOWED FOR THIS PASSWORD.
*
             SPACE
PFOUND       DS      0H
             SR      R9,R9                  CLEAR REG 9
             IC      R9,TRANCNUM            GET NUMBER OF TRANCODES FOR
PASSWRD
             L       R6,TRANCTAB-1          GET TRANCODE TABLE ADDRESS
```

Figure B-15 Sample Input Exit IXXUIN (Page 2 of 4)

```
             L       R5,ATRANCDE            GET TRANCODE FROM INPUT
LOOPTCDE     DS      0H
             CLC     TRANCODE,0(R8)         COMPARE INPUT / TABLE
             BE      PASSWOK                STOP LOOKING IF EQUAL
             LA      R6,L'TRANCODE(R6)      GET NEXT TABLE ENTRY
             BCT     R9,LOOPTCDE            LOOP BACK
             B       TNOTFND                PASSWORD IS INVALID
             SPACE   3
*
* DELETE MODULE, IF LOADED, BEFORE RETURNING TO SYSTEM
*
             SPACE
PASSWOK      DS      0H
             TM      ATABLE,X'80'           WAS MODULE LOADED?
             BZ      RETURN                 NO, BYPASS
*
             EXEC    CICS RELEASE PROGRAM ('IXXSTABL')
*
             SPACE   5
RETURN       DS      0H
             EJECT
             SPACE   25
*****************************************************************
*                                                               *
*             ADDITIONAL CODE CAN BE APPENDED HERE              *
*                                                               *
*               PLEASE BEWARE OF DUPLICATE LABELS               *
*                                                               *
*****************************************************************
             EJECT
*
* RETURN TO CALLER
*
             SPACE
             DFHEIRET
             SPACE   3
*
* ERROR ROUTINES
*
             SPACE
PNOTFND      DS      0H
             MVC     MSGLENTH,=AL2(LERRMSG1)
             MVC     MESSAGE(LERRMSG1),ERRMSG1
             B       RETURN
             SPACE   2
PLNTHERR     DS      0H
             MVC     MSGLENTH,=AL2(LERRMSG2)
             MVC     MESSAGE(LERRMSG2),ERRMSG2
             B       RETURN
             SPACE   2
NOTFOUND     DS      0H
             MVC     MSGLENTH,=AL2(LERRMSG3)
             MVC     MESSAGE(LERRMSG3),ERRMSG3
             B       RETURN
             SPACE   2
TNOTFND      DS      0H
             MVC     MSGLENTH,=AL2(LERRMSG4)
             MVC     MESSAGE(LERRMSG4),ERRMSG4
             B       RETURN
             SPACE   3
*
* REMOTE INSTRUCTIONS
*
             SPACE
MVCPASS      MVC     PASSWORD(0),0(R5)
MVCBLNK      MVC     0(0,R5),=CL8' '
             EJECT
*
* AREAS AND CONSTANTS
*
             SPACE
```

Figure B-15 Sample Input Exit IXXUIN (Page 3 of 4)

```
ATABLE        DC      V(IXXSTABL)            SECURITY TABLE ADDRESS, IN
LINKED
PTABADDR      DS      A
ABEGIN        DS      A
PASSWORD      DC      CL8' '
*
ERRMSG1       DC      C'IXXSEC01 PASSWORD NOT FOUND IN INPUT MESSAGE;;'
ERRMSG2       DC      C'IXXSEC02 PASSWORD EXCEEDS 8 CHARACTERS;;'
ERRMSG3       DC      C'IXXSEC03 SECURITY VIOLATION;;'
ERRMSG4       DC      C'IXXSEC04 SECURITY VIOLATION;;'
LERRMSG1      EQU     L'ERRMSG1
LERRMSG2      EQU     L'ERRMSG2
LERRMSG3      EQU     L'ERRMSG3
LERRMSG4      EQU     L'ERRMSG4
              SPACE   5
              LTORG
              EJECT
PASSWINF      DSECT
PASSWTAD      DS      A                      PASSWORD TABLE ADDRESS
PASSWNUM      DS      H                      NUMBER OF PASSWORDS
PASSWTLN      DS      H                      TABLE ELEMENT LENGTH
              SPACE   3
PASSWTAB      DSECT
PASSW         DS      CL8                    PASSWORD
TRANCNUM      DS      X                      NUMBER OF TRANSACTION CODES
*                                              ASSOCIATED WITH THIS
PASSWORD
TRANCTAB      DS      AL3                    TRANCODE TABLE ADDRESS
*                                              FOR THIS PASSWORD
PASSWEND      EQU     *
PASSWELN      EQU     PASSWEND-PASSWTAB      LENGTH OF TABLE ELEMENT
              SPACE   3
TRANTABL      DSECT
TRANCODE      DS      CL8                    TRANSACTION CODE
              SPACE   5
PARMLIST      DSECT
APSB          DS      A                      PSB ADDRESS
AEXIT         DS      A                      EXIT ACTION WORD ADDRESS
AINPUT        DS      A                      INPUT MESSAGE ADDRESS
ATRANCDE      DS      A                      TRANSACTION CODE ADDRESS
ALTERM        DS      A                      LTERM ADDRESS
              SPACE   5
INPUTMSG      DSECT
MSGLENTH      DS      H                      MESSAGE LENGTH
              DS      H
MESSAGE       DS      CL4000                 MESSAGE
              SPACE   5
              IXUSER
              SPACE   5
              END
```

Figure B-15 Sample Input Exit IXXUIN (Page 4 of 4)

Invoking the IXXUIN Input Exit

Inquiries processed by the sample input exit must provide a password as the first item of the inquiry. The password specified is a one to eight character string that must be specified in the first 80 bytes of the inquiry.

Figure B-16 illustrates the specifications of the password in the inquiry.

```
PAGE P/N TRANSACTION: IISY INQUIRY:
                      Enter Inquiry Below:
SYSTEM DISPLAY PLANT PLANT.ID PLANT.NAME ;


 PLANT.ID  PLANT.NAME
 10100     DALLAS SALES
 20150     REMOTE CONTROL PRODUCTS
 30200     CORPORATE HDQTRS DALLAS
 40300     DELUXE PRODUCTS
 50300     MECHANICAL PRODUCTS
 60200     BASIC TOYS
 70500     DISTRIBUTION
 IXX9121 END OF INQUIRY              (8,7 USER DB CALLS,ROOTS)
```

Figure B-16 Inquiry with an Input Exit

Where:

IISY	Is the transaction identifier (application name)
SYSTEM	Is the password

Input Exit Security Table (IXXSTABL)

The security table, internally named, is a load module compiled and link edited separately. It can be link edited with the exit, making it resident, or it can be loaded dynamically during execution. The link edited load module name must be IXXSTABL.

Figure B-17 illustrates the generation of the password table with the IXSECTY and IXSGEN macros.

```
        PRINT NOGEN
SECT    TITLE '*** INPUT EXIT: SECURITY ROUTINE TABLE EXAMPLE ***'
        IXSECTY PSSWORD=PRO,TRANCDE=IIPR
        SPACE 5
        IXSECTY PSSWORD=SYSTEM,TRANCDE=(IIPR,IISY,IITS)
        SPACE 5
        IXSECTY PSSWORD=TEST,TRANCDE=(IITS)
        SPACE 5
        IXSGEN
        SPACE 5
        END
```

Figure B-17 Specifying Passwords and Transactions with the IXSECTY and IXSGEN Macros

Input Exit Error Messages

Whenever the input exit detects an unexpected error, the inquiry in error is replaced by an error message. This in turn causes a syntax error and the invalid inquiry is rejected by VISION:Inquiry.

The error messages issued by the IXXUIN exit are the following:

```
IXXSEC01 PASSWORD NOT FOUND IN INPUT MESSAGE
```

Issued when the password is not specified within the first 80 bytes of the inquiry.

```
IXXSEC02 PASSWORD EXCEEDS 8 CHARACTERS
```

Self-explanatory.

```
IXXSEC03 SECURITY VIOLATION
```

Issued when the password is not found in the security table.

```
IXXSEC04 SECURITY VIOLATION
```

Issued when the transaction code is not valid for the specified password.

Figure B-18 illustrates an inquiry containing security violations in which error messages are issued.

```
DISPLAY PLANT PLANT.ID PLANT.NAME;;

IXX0106 DATA NAME 'IXXSEC03' NOT FOUND IN DIRECTORY
IXX0106 DATA NAME 'SECURITY' NOT FOUND IN DIRECTORY
IXX0106 DATA NAME 'VIOLATION' NOT FOUND IN DIRECTORY
IXX0108 COMMAND NOT FOUND IN INQUIRY
```

Figure B-18 An Inquiry Containing Security Violations Determined by the Input Exit Routine

Implementing the Sample Input Exit

Perform the following steps to implement the input exit:

1. Assemble IXXUIN, the sample input exit.
2. Include the object module of the IXXUIN Input exit routine into an SMP/E USERMOD and apply them to the VISION:Inquiry load module. See Chapter 7, "User Exits" for an example of a USERMOD for a user exit and the sample SMP/E JCL.
3. Define the macro library containing the IXSECTY and IXSGEN macros in the SYSLIB DD statement and assemble and link-edit the IXXSTABL security table. This will create a standalone security load module that can be loaded dynamically.

Sample Sort Exit (IXCUSRTS)

The source library members II.TCVLSRC (IXCUSRTS) is a sample online sort exit written in Assembler language. It can be implemented in its entirety, or it can be used as a guide for designing and coding your own sort exit with your specific installation requirements in mind.

The purpose of this exit is to start a new transaction (IQDV in this sample) which deletes the output records from the VSAM work data set after the sort processing.

Figure B-19 contains the source code for the sample sort exit, IXCUSRTS.

```
SORTEXIT TITLE 'ONLINE SORT USER EXIT - IXCUSRT'
*
*  IN THE FOLLOWING SAMPLE ROUTINE:
*
*        1- REGISTER 8 POINTS TO THE EXEC INTERFACE BLOCK(EIB).
*        2- THE TRANSACTION ID, IQDV, CAN BE MODIFIED AND MUST BE
*           DEFINED IN THE PCT TABLE OF CICS.
*        3- THE CICS TERMINAL ID WILL BE PASSED TO THE DELETE ROUTINE
*           STARTED BY TRANSACTION IQDV WHICH WILL BE USED AS THE
*           FIRST FOUR CHARACTERS OF THE KEY FIELD FOR GENERIC DELETE.
*
*
IXCUSRT  DFHEIENT EIBREG=8
         EXEC CICS ADDRESS EIB(8)
         EXEC CICS START TRANSID('IQDV') FROM(EIBTRMID) LENGTH(4)
         END
```

Figure B-19 Sample Online Sort Exit IXCUSRTS

Figure B-20 contains the source code for the sample delete routine, II.TCVLSRC (IXCUSDL), associated with transaction, IQDV, which deletes the records from the VSAM work data set. Note that the generic key used for deletion of the output records is the CICS terminal ID which is passed to the delete routine by the sort exit routine, IXCUSRT.

```
SORTDEL  TITLE 'DELETES SORT RECORDS FOR VSAM WORK FILE'
*
*  - THIS IS A STANDALONE DELETE PROGRAM ASSOCIATED WITH THE
*    TRANSACTION IQDV WHICH WILL BE STARTED BY THE SORT EXIT
*    ROUTINE IXCUSRT.
*  - THE ENTRY FOR THIS ROUTINE MUST BE DEFINED IN THE PCT AND PPT
*    TABLES OF CICS.
*
DFHEISTG DSECT
         DFHEISTG
IO_LEN   DS    0H                USED TO KEEP RETRIEVE HAPPY
ENQ_NAME DS    2CL4              USED TO BUILD ENQ-NAME
EIEND#X  DS    0D                FORCE ALIGNMENT
IXCUSDL  DFHEIENT
*        *     **** FIND OUT WHAT TERMINAL THIS REQUEST IS FOR ****
         EXEC  CICS RETRIEVE SET(10) LENGTH(IO_LEN)
         MVC   ENQ_NAME,=C'SORT'       BUILD ENQNAME
         MVC   ENQ_NAME+4,0(10)        WITH TERMINAL NAME AS SUFFIX
*        *     **** ENQUEUE RESOURCE ****
         EXEC  CICS ENQ RESOURCE(ENQ_NAME) LENGTH(8)
*        *     **** DELETE RECORDS CREATED FOR THAT TERMINAL ****
         EXEC  CICS DELETE DATASET('SORTWORK') RIDFLD(0(10)) GENERIC   -
               KEYLENGTH(4) NOHANDLE
*        *     **** AUTOMATIC DEQ BY CICS AT THE END ****
         END
```

Figure B-20 Sample Delete Routine IXCUSDL Started by the Sort Exit IXCUSRT

Implementing the Sample Sort Exit

To implement the sample sort exit:

1. Assemble the sample sort exit routine IXCUSRTS.
2. Include the object module of the sort exit routine into an SMP/E USERMOD and apply them to the VISION:Inquiry load module. See Chapter 7, "User Exits" for an example of a USERMOD for a user exit and the sample SMP/E JCL.

   Note: This exit can only be used when the SORTIOM parameter is selected as VS in the CICS parameter module II.TCVLSRC (CVLCPARM).
3. You must link edit the IXCUSDL routine which is associated with transaction IQDV separately and have the appropriate program and transaction entries in your CICS system.

C The IXSECTY and IXSGEN Macros

The IXSECTY and IXSGEN macros are provided for the creation of a password table and are capable of defining a table with a capacity of up to 100 passwords and 500 transaction codes. If these maximums are insufficient for your installation needs, globals in the macros can be modified accordingly.

IXSECTY Macro

One IXSECTY macro must be defined for each password in the table. The format of the IXSECTY macro is:

LABEL	OPERATION	OPERAND
	IXSECTY	PSSWORD=name ,TRANCDE=name | (name$_1$, . . . ,name$_n$)

The IXSECTY operands are:

PSSWORD=name
Name is a unique password that is to be associated with one or more transaction codes, defined by the TRANCDE parameter.

TRANCDE=name | (name$_1$, . . . ,name$_n$)
Name is the transaction code to be associated with the password defined by the PSSWORD parameter.

More than one transaction code can be defined for one particular password. The names must be enclosed in parentheses, separated by commas.

The following illustrates typical specifications of this macro.

```
IXSECTY PSSWORD=PRO,TRANCDE=IIPR
IXSECTY PSSWORD=SYSTEM,TRANCDE=(IIPR,IISY,IITS)
IXSECTY PSSWORD=TEST,TRANCDE=(IITS)
```

The following figure contains the source code for the IXSECTY macro. This macro should be added to your appropriate installation macro library.

```
             MACRO
             IXSECTY &PSSWORD=,
                     &TRANCDE=
             GBLA    &PASSN,&TRANN(100),&PSEQ(100),&TRANCNT,&TRANP(500)
.*
             GBLB    &SECEND,&ERRFLAG
.*
             GBLC    &PASSW(100),&TRANCODE(500)
.*
             LCLA   &I,&J,&K,&L,&M,&N,&O
             LCLC   &FILLER
.*
             AIF    (T'&PSSWORD NE '0').PASSWOK
&ERRFLAG     SETB   1
             MNOTE  12,'PASSWORD PARAMETER OMITTED, MACRO IGNORED'
.*
.PASSWOK     ANOP
             AIF    (T'&TRANCDE NE '0').TRANCOK
&ERRFLAG     SETB   1
             MNOTE  12,'TRANCODE PARAMETER OMITTED, MACRO IGNORED'
.*
.TRANCOK     ANOP
&O           SETA   K'&PSSWORD
             AIF    (&0 LE 8).LNTHPOK
&ERRFLAG     SETB   1
             MNOTE  12,'PASSWORD LENGTH GT 8, MACRO IGNORED'
.*
.LNTHPOK     ANOP
&N           SETA   N'&TRANCDE
.LOOPLTH     ANOP
&K           SETA   &K+1
             AIF    (&K GT &N).OLOOPL
             AIF    (K'&TRANCDE(&K) LE 4).LOOPLTH
&ERRFLAG     SETB   1
             MNOTE  12,'TRANCODE &TRANCDE(&K) LENGTH GT 4, MACRO IGNORED'
             AGO    .LOOPLTH
.OLOOPL      ANOP
.*
             AIF    (NOT &SECEND).POSOK
&ERRFLAG     SETB   1
             MNOTE  12,'IXSECTY MACRO ISSUED AFTER IXSGEN, IGNORED'
.*
.POSOK       ANOP
&L           SETA   &L+1
             AIF    (&L GT &PASSN).OLOOPD
             AIF    ('&PASSW(&L)' NE '&PSSWORD').POSOK
&ERRFLAG     SETB   1
             MNOTE  12,'DUPLICATE PASSWORD SPECIFIED, MACRO IGNORED'
             AGO    .POSOK
 OLOOPD      ANOP
.*
             AIF     (&ERRFLAG).OUTMAC
.*
&FILLER      SETC    ''
&O           SETA    8-&O
             AIF     (&O EQ 0).NOFILL
&FILLER      SETC    '      '(1,&O)
.NOFILL      ANOP
.*
&PASSN       SETA    &PASSN+1              NUMBER OF PASSWORDS + 1
&TRANN(&PASSN) SETA   &N                   NUMBER OF TRANCODES PER PASSWORD
&PASSW(&PASSN) SETC   `&PSSWORD'.'&FILLER' PASSWORD
&PSEQ(&PASSN) SETA &PASSN                  PASSWORD SEQUENCE NUMBER
```

Figure C-1 IXSECTY Macro Source Code (Page 1 of 2)

```
.*
&TRANCNT      SETA    &TRANCNT+1
&TRANCODE(&TRANCNT) SETC  '&TRANCDE(&J)'
&TRANP(&TRANCNT) SETA &PASSN
              AGO     .LOOP2
.*
.OLOOP2       ANOP
.OUTMAC       MEXIT
              MEND
```

Figure C-1 IXSECTY Macro Source Code (Page 2 of 2)

IXSGEN Macro

The IXSGEN macro is specified once for each table generation. It must follow the last IXSECTY macro, and it must be followed by an Assembler END statement.

The IXSGEN macro consists of an operator and no operands. The format of the IXSGEN macro is:

LABEL	OPERATION	OPERAND
	IXSGEN	

The following figure contains the source code for the IXSGEN macro. Add this macro to your installation macro library.

```
              MACRO
              IXSGEN
.*
              GBLA    &PASSN,&TRANN(100),&PSEQ(100),&TRANCNT,&TRANP(500)
              GBLB    &SECEND,&ERRFLAG
              GBLC    &PASSW(100),&TRANCODE(500)
.*
              LCLA    &I,&J,&K,&L,&M,&N,&O,&P
              LCLB    SWITCH
              LCLC    &PASSWRD
.*
              AIF     (NOT &ERRFLAG).GEN
              MNOTE   12,'ONE OR MORE IXSECTY MACRO(S) IN ERROR, IGNORED'
              MEXIT
.*
.GEN          ANOP
              AIF     (NOT &SECEND).GENOK
              MNOTE   12,'MORE THAN ONE IXSGEN MACRO SPECIFIED, IGNORED'
              MEXIT
.*
.GENOK        ANOP
&SECEND       SETB    1
.*
*MACRO        ISXGEN -                                          - V.1.0
              SPACE
IXXSTABL      CSECT
              SPACE   5
              DC      A(PASSWT)              PASSWORD TABLE ADDRESS
              DC      AL2(&PASSN,PASSWELN)   NUMBER OF PASSWORDS
.*                                           LENGTH OF TABLE ELEMENT
              SPACE   5
```

Figure C-2 IXSGEN Macro Source Code (Page 1 of 3)

```
                CNOP    0,4
PASSWT          EQU     *
.*
.SORT           ANOP
                AIF     (&PASSN EQ 1).LOOPP
&N              SETA    &PASSN-1
&SWITCH         SETB    0
&K              SETA    0
.*
.SORT1          ANOP
&k              SETA    &K+1
                AIF     (&K GT &N).SORT2
&L              SETA    &K+1
                AIF     ('&PASSW(&K)' LE '&PASSW(&L)').SORT1
&SWITCH         SETB    1
&PASSWRD        SETC    '&PASSW(&K)'
&PASSW(&K)      SETC    '&PASSW(&L)'
&PASSW(&L)      SETC    '&PASSWRD'
&O              SETA    &TRANN(&K)
&TRANN(&K)      SETA    &TRANN(&L)
&TRANN(&L)      SETA    &O
&P              SETA    &PSEQ(&K)
&PSEQ(&K)       SETA    &PSEQ(&L)
&PSEQ(&L)       SETA    &P
                AGO     .SORT1
.*
.SORT2          ANOP
                AIF     (&SWITCH).SORT
.*
.LOOPP          ANOP
&M              SETA    &M+1
                AIF     (&M GT &PASSN).OLOOPP
                DC      CL8'&PASSW(&M)'    PASSWORD
                DC      AL1(&TRANN(&M))    NUMBER OF TRANCODES ASSOCIATED WITH
.*                                             THIS PASSWORD
                DC      AL3(TRAN&PSEQ(&M)) TRANCODE TABLE ADDRESS FOR
.*                                             THIS PASSWORD
                SPACE   5
                AGO     .LOOPP
.*
.OLOOPP         ANOP
                SPACE   10
.LOOPT          ANOP
&I              SETA    &I+1
                AIF     (&I GT &TRANCNT).OLOOPT
                AIF     (&TRANP(&I) EQ &J).SAMEP
                SPACE   3
TRAN&TRANP(&I)  DS      0CL8
&J              SETA    &TRANP(&I)
.SAMEP          ANOP
                DC      CL8'&TRANCODE(&I)'
                AGO     .LOOPT
.*
.OLOOPT         ANOP
                SPACE   10
PASSWINF        DSECT
PASSWTAD        DS      A                  PASSWORD TABLE ADDRESS
PASSWNUM        DS      H                  NUMBER OF PASSWORDS
PASSWTLN        DS      H                  TABLE OF ELEMENT LENGTH
                SPACE   3
PASSWTAB        DSECT
PASSW           DS      CL8                PASSWORD
TRANCNUM        DS      X                  NUMBER OF TRANSACTION CODES
.*                                            ASSOCIATED WITH THIS PASSWORD
TRANCTAB        DS      AL3                TRANCODE TABLE ADDRESS
.*                                            FOR THIS PASSWORD
PASSWEND        EQU     *
PASSWELN        EQU     PASSWEND-PASSWTAB      LENGTH OF TABLE ELEMENT
                SPACE   3
TRANTABL        DSECT
TRANCODE        DS      CL8                    TRANSACTION CODE
                SPACE   5
PARMLIST        DSECT
```

Figure C-2 IXSGEN Macro Source Code (Page 2 of 3)

```
APSB          DS      A                   PSB ADDRESS
AEXIT         DS      A                   EXIT ACTION WORD ADDRESS
AINPUT        DS      A                   INPUT MESSAGE ADDRESS
ATRANCDE      DS      A                   TRANSACTION CODE ADDRESS
ALTERM        DS      A                   TERM ADDRESS
              SPACE   5
INPUTMSG      DSECT
MSGLENTH      DS      H                   MESSAGE LENGTH
              DS      H
MESSAGE       DS      CL2000              MESSAGE
              SPACE   5
              MEXIT
              MEND
```

Figure C-2 IXSGEN Macro Source Code (Page 3 of 3)

D System Modules

The complete VISION:Inquiry system consists of the following modules:

- Native VISION:Inquiry modules
- Text Editor facility modules
- AQF modules
- VISION:Journey modules.
- Intraccess module

The storage requirements for the system includes storage for the load modules, I/O buffers, PL/I transients, VISION:Inquiry working storage, and IMS.

Native VISION:Inquiry Modules

The native VISION:Inquiry portion of the system consists of the following programs which reside in the installation load module library for VISION:Inquiry.

Load Module Name	Execution Environment	Programming Language
IXXSTRT	Online	PL/I and Assembler
IXCSORT	Online	Assembler
INQIO	Online	Assembler
CVLCPARM	Online	Assembler
INQMAP1M	BMS Map	Assembler
INQMAP2M	BMS Map	Assembler
IIBATCH	DL/I Batch or Batch	PL/I and Assembler
IIDEMO	DL/I Batch	PL/I
IVDEMO	Batch	PL/I

Load Module Name	Execution Environment	Programming Language
IIGEN DL/I	Batch or Batch	PL/I
IIINIT	DL/I Batch or Batch	PL/I
IXUSTAT	DL/I Batch or Batch	PL/I
IXUUNLD	DL/I Batch or Batch	PL/I
IXULOAD	DL/I Batch or Batch	PL/I
IXUSQRY	DL/I Batch or Batch	PL/I
IXUIQRY	DL/I Batch or Batch	PL/I
IXXRMODL	DL/I Randomizer (used only for DL/I System Database)	Assembler
CUYSHDG	DL/I Batch or Batch	Assembler
CUYGCHK	DL/I Batch or Batch	Assembler
CUYSHMG	DL/I Batch, Batch, or Online	Assembler

Text Editor Modules

The Text Editor consists of the following executable modules and maps which reside in the installation load module library.

Load Module Name	Execution Environment	Programming Language
INQEDIT	Online	PL/I
IQEMAPMM	BMS Map	Assembler
IQEMAPHM	BMS Map	Assembler
CVLCPARM	Online	Assembler
CUYXEMSG	Online	Assembler

AQF Modules

The AQF portion of the system consists of the following executable programs and maps (menus) which reside in the installation load module library.

Load Module Name	Execution Environment	Programming Language
IDBEH01M	Online (BMS Map)	Assembler
IDBEH02M	Online (BMS Map)	Assembler
IDBEH03M	Online (BMS Map)	Assembler
IDBEH04M	Online (BMS Map)	Assembler
IDBEH05M	Online (BMS Map)	Assembler
IDBEH06M	Online (BMS Map)	Assembler
IDBEH07M	Online (BMS Map)	Assembler
IDBEH10M	Online (BMS Map)	Assembler
IDBEH13M	Online (BMS Map)	Assembler
IDBEM01M	Online (BMS Map)	Assembler
IDBEM02M	Online (BMS Map)	Assembler
IDBEM03M	Online (BMS Map)	Assembler
IDBEM04M	Online (BMS Map)	Assembler
IDBEM05M	Online (BMS Map)	Assembler
IDBEM06M	Online (BMS Map)	Assembler
IDBEM07M	Online (BMS Map)	Assembler
IDBEM10M	Online (BMS Map)	Assembler
IDBEM13M	Online (BMS Map)	Assembler
IDBERRSM	Online (BMS Map)	Assembler
IDB101	Online	PL/I
IDB102	Online	PL/I
IDB104	Online	PL/I
IDB105	Online	PL/I

Load Module Name	Execution Environment	Programming Language
IDB106	Online	PL/I
IDB107	Online	PL/I
IDB108	Online	PL/I
IDB109	Online	PL/I
IDB110	Online	PL/I
IDB111	Online	PL/I
IDB112	Online	PL/I
IDB113	Online	PL/I
IDB114	Online	PL/I
CVLCAMOD	Online	Assembler
CVLCAJPR	Online	Assembler
CVLCPARM	Online	Assembler

DB2 Modules

The DB2 modules are linked with the other VISION:Inquiry executable load modules when the DB2 option of VISION:Inquiry is available on your system. There is only one standalone DB2 load module, DIOSQLC, that needs to be present in the load library to access DB2 tables.

Load Module Name	Execution Environment	Programming Language
DIOSQLC	Batch or Online	Assembler

VISION:Journey Modules

The VISION:Journey facility consists of the following executable load modules stored in the installation program library.

Load Module Name	Execution Environment	Programming Language
DYLC0SS	Online	Assembler
DYLC010	Online	Assembler
DYLC020	Online	Assembler
DYLC030	Online	SAS/C
IFUCLEN	Batch	PL/I
IDFTSP7M	Online (BMS Map)	Assembler

Intraccess Module

VISION:Inquiry uses the following load module that resides in the installation load module library when it processes the Intraccess queries. For the other load modules specific to Intraccess, see the Intraccess documentation.

Load Module Name	Execution Environment	Programming Language
INQINTRA	Online	Assembler

E VISION:Inquiry Target and Distribution Libraries

This appendix contains the information about the content of the VISION:Inquiry target and distribution libraries. It also lists the members in the source and control libraries with a brief description about each member.

The following table shows the default name of the target and distribution libraries:

Target Library Name	Distribution Library Name	Contains
II.TCVLCNTL	II.DCVLCNTL	Sample JCLs for Post-Installation and maintenance of the system
II.TCVLMAC	II.DCVLMAC	Macros for customization, exit routines, and building BMS screens
II.TCVLPGM	II.DCVLPGM	VISION:Inquiry load modules
II.TCVLSRC	II.DCVLSRC	Source for various control blocks, data for test systems, source for BMS screens, and error messages, and vocabulary
II.TCVLOBJ	II.DCVLOBJ	VISION:Inquiry object modules in the link-edited format
II.TCVLCLST	II.DCVLCLST	CLISTs for Post-Installation Dialog
II.TCVLMLIB	II.DCVLMLIB	Messages for Post-Installation Dialog
II.TCVLPLIB	II.DCVLPLIB	Panels for Post-Installation Dialog
II.TCVLSLIB	II.DCVLSLIB	Skeleton JCLs for Post-Installation Dialog

Source Library

The source library contains the following members

CIPSB	Sample PSB for executing inquiries against the test databases
CIPSB	AQF Sample PSB for AQF accessing DL/I system database
CUYGCHK	IIGEN checkpoint interval control module
CUYSHDG	Utility program headings
CUYSHMG	Hard-coded messages
CUYXEMSG	Text Editor error messages
CVLCAJPR	AQF job submission parameters
CVLCAMOD	AQF BMS mapset names
CVLCPARM	VISION:Inquiry online processing parameters
DB2CATA	DB2 catalog program
DJCSECT	VISION:Journey load module names and download data set ddname CSECT
IDBEHnn IDBEMnn IDBERRS	Source for generating BMS maps for AQF
IDFTSP7	Source for generating BMS map for VISION:Journey
IIDATA	Data for loading the IMS (DL/I) test databases, PLANT and SKILL
IIDBDDM	Sample DBD for the PLANT test database
IIDBDDS	Sample DBD for the SKILL test database
IIDMGEN	Sample IIGEN statements that describe all test databases
IIERROR	Standard system error messages
IIPSB	Sample PSB for inquiries against the test IMS (DL/I) databases
IIPSB01	Sample PSB for initializing an IMS (DL/I) system database using the IIINIT utility
IIPSB02	PSB for maintaining an IMS (DL/I) system database using the IIGEN, IXUIQRY, IXULOAD, IXUSQRY, IXUSTAT, and IXUUNLD utilities

IIPSB03	Sample PSB for loading the IMS (DL/I) test databases using IIDEMO
IIVOCAB	Standard system vocabulary
INQIO	VISION:Inquiry front-end program in CICS (online processing)
INQMAP1 INQMAP2	Source for generating BMS maps for native VISION:Inquiry
IQEMAPH IQEMAPM	Source for generating BMS maps for Text Editor facility (main panel and help panel)
IXBIDENT	System identification module for the batch inquiry program
IXCUSDL	Online delete sample user exit (invoked by IXCUSRT)
IXCUSECS	Online security sample user exit
IXCUSRTS	Online sort sample user exit
IXSIDENT	System identification module for the online inquiry program and with VISION:Journey
IXXDB	Sample DBD for IMS (DL/I) system database
IXXUCONP	Sample user conversion exit in PL/I
IXXUCONS	Sample user conversion exit in Assembler
IXXUFNCS	Sample user function exit
IXXUINS	Sample user input exit
IXXUOUTS	Sample user output exit
IXXUSECS	Sample user security exit
VSDATAF	Data used to initialize the VISION:Journey VSAM download data set
VSDATAH VSDATAK VSDATAR	Data used to load the VSAM test data sets
VSDATAS	Data used to initialize the sort work data set

Control Library

The control library contains the following members

ASMCAJPR	JCL to assemble the AQF job submission parameters, CVLCAJPR
ASMCAMOD	JCL to assemble the AQF BMS mapset names module, CVLCAMOD
ASMCHKI	JCL to assemble the IIGEN checkpoint interval control module, CUYGCHK
ASMCPARM	JCL to assemble the VISION:Inquiry online processing parameters, CVLCPARM
ASMEMSG	JCL to assemble the Text Editor messages, CUYXEMSG
ASMHMSG	JCL to assemble the hard-coded messages module, CUYSHMG
ASMINQIO	JCL to assemble the VISION:Inquiry front-end program, INQIO
ASMSHDG	JCL to assemble the utility headings module, CUYSHDG
CICSJCL	Sample JCL to be included in CICS startup JCL
DB2BIND	TSO commands to bind the DB2 plan
DB2CATL	JCL to run the DB2 catalog program DB2CATA
DB2CREAT	SPUFI input to create and index a DB2 system database
DB2DEMO	SPUFI input to create and load the DB2 test tables and views
DB2ELEM	JCL to define the test tables/views to a DB2 system database using the IIGEN utility
DB2INDEX	SPUFI input to create the index for DB2 test tables
DB2INIT	JCL to initialize a DB2 system database using the IIINIT utility
DB2IQRY	JCL to convert stored inquiries and functions for a DB2 system database using the IXUIQRY utility
DB2LOAD	JCL to reload a DB2 system database using the IXULOAD utility
DB2SQRY	JCL to unload stored inquiries and functions from a DB2 system database using the IXUSQRY utility

DB2STAT	JCL to execute the IXUSTAT utility for a DB2 system database
DB2UNLD	JCL to unload a DB2 system database using the IXUUNLD utility
IIPSBDBD	JCL for DBD, PSB, and ACB generations
IMSCOBL	JCL to run COBOL converter program for IMS (DL/I) databases
IMSDEMO	JCL to allocate and load the IMS (DL/I) test databases, PLANT and SKILL
IMSELEM	JCL to define the test databases to an IMS (DL/I) system database using the IIGEN utility
IMSINIT	JCL to allocate and initialize an IMS (DL/I) system database using the IIINIT utility
IMSIQRY	JCL to convert stored inquiries and functions for an IMS (DL/I) system database using the IXUIQRY utility
IMSLOAD	JCL to reload an IMS (DL/I) system database using the IXULOAD utility
IMSSQRY	JCL to unload stored inquiries and functions from an IMS (DL/I) system database using the IXUSQRY utility
IMSSTAT	JCL to execute the IXUSTAT utility for an IMS (DL/I) system database
IMSUNLD	JCL to unload an IMS (DL/I) system database using the IXUUNLD utility
INQCSDUP	JCL to define VISION:Inquiry entries to the CICS System Definition (CSD) file using DFHCSDUP utility of CICS
IQDB2CSD	JCL to define the DB2 option entries of the VISION:Inquiry to the CICS System Definition (CSD) file using DFHCSDUP utility of CICS
IQBATD	JCL to run batch inquiries in IMS (DL/I) environment
IQBATD 2	JCL to run batch inquiries to access DB2 tables in IMS (DL/I) environment
IQBATV	JCL to run batch inquiries in non-IMS (DL/I) environment
MAPASM	JCL and PROC to build BMS maps
VSAMCOBL	Sample JCL to run COBOL converter program for VSAM data sets

VSAMDEMO	JCL to allocate and load the VSAM test data sets, VSPLANT and VSSKILL
VSAMELEM	JCL to define the test databases in a VSAM system database using the IIGEN utility
VSAMINIT	JCL to allocate and initialize a VSAM system database using the IIINIT utility
VSAMIQRY	JCL to execute the IXUIQRY utility for a VSAM system database
VSAMLOAD	JCL to reload a VSAM system database using the IXULOAD utility
VSAMSQRY	JCL to execute the IXUSQRY utility for a VSAM system database
VSAMSTAT	JCL to execute IXUSTAT utility for a VSAM system database
VSAMUNLD	JCL to unload a VSAM system database using the IXUUNLD utility
VSMFTSCL	Sample JCL to execute the IFUCLEN cleanup utility for the VISION:Journey VSAM download data set
VSMFTSIN	Sample JCL to allocate and initialize the VISION:Journey VSAM download data set
VSMFTSRE	JCL to reorganize the VISION:Journey VSAM download data set to eliminate the CI/CA (Control Interval/Control Area) splits
VSMHDEMO	JCL to allocate and load the VSAM test data sets, VSHPLANT and VSHSKILL
VSMSRTIN	Sample JCL to allocate and initialize the VSAM sort work data set VS.SORTDS

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W