Advantage™ VISION:Inquiry® for IMS™ and TSO

Technical Reference Guide

6.5

Computer Associates®

IQRFI065.PDF/D07-202-010

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, the user may print a reasonable number of copies of this documentation for its own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software of the user will have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2003 Computer Associates International, Inc. (CA).

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introducing VISION:Inquiry

CD-ROM Contents 1-1
- About the Online Documentation 1-1
- Installing Online Documentation and the Acrobat Reader 1-1
- Viewing Online Documentation 1-2
- Using Adobe Acrobat Reader 1-2

Contacting Total License Care (TLC) 1-2
Contacting Computer Associates 1-3
Product Description 1-3
DB2 Information 1-4
Intraccess Information 1-4
Objectives of this Guide 1-4
How this Guide Is Organized 1-5
VISION:Inquiry Configuration Requirements 1-6
Product Terminology 1-6
Associated Documents 1-7

Chapter 2: System Overview

VISION:Inquiry Features and Capabilities 2-2
- VISION:Inquiry Processing Capabilities 2-2
- VISION:Inquiry Display Capabilities 2-3
- VISION:Journey for Windows Capabilities 2-3
- Intraccess Capabilities 2-3

VISION:Inquiry System Components 2-4
- The VISION:Inquiry System Database 2-4
- The Inquiry Processing Programs 2-5

VISION:Inquiry System Flow 2-6
VISION:Inquiry MPP Processing 2-7
VISION:Inquiry BMP Processing 2-8
VISION:Inquiry Batch Processing 2-9
VISION:Inquiry TSO Processing 2-10
VISION:Inquiry Text Editor Processing 2-11
VISION:Inquiry AQF Processing 2-11
VISION:Journey Processing 2-11
Intraccess Processing 2-12
VISION:Inquiry Native SQL Syntax Support 2-12
User Exits 2-12
The VISION:Inquiry Utilities 2-14
IIINIT Utility 2-14
IIGEN Utility 2-14
IXUUNLD Utility 2-15
IXULOAD Utility 2-15
IXUSQRY Utility 2-15
IXUIQRY Utility 2-16
IXUSTAT Utility 2-16
IAOINIT Utility 2-16
IFUINIT Utility 2-16
IFUCLEN Utility 2-16

Chapter 3: System Components

The System Database 3-1
System Database Index 3-3
Database Directory Block 3-5
The System Vocabulary 3-5
VISION:Inquiry Messages 3-6
Database Maps 3-6
User Directories 3-7
Connected Directories 3-8
Terminal Descriptions (LTERMs) 3-9
Conversational Scratch Pad Storage 3-10
Automatic Query Facility (AQF) Components 3-10
VISION:Journey Download Database 3-11
Text Editor Work Database 3-11
VISION:Inquiry Test Data 3-12
IMS (DL/I) Test Databases 3-12
DB2 Test Tables 3-13
DB2 Test Views 3-14
VSAM Non-Hierarchical Test Data Sets 3-14
VSAM Hierarchical Test Data Sets 3-15

Chapter 4: The Definition Process

Defining the Directory 4-2
DIRECTORY Control Statement Group 4-2
Eliminating Fields and Vocabulary 4-3
Defining VISION:Inquiry 4-3
Using the IIGEN Utility 4-3
IIGEN Functions 4-3
IIGEN Control Statement Groups 4-5
IIGEN Names 4-5
IIGEN Coding Rules 4-6
Notation 4-6
Defining the Application 4-7
APPL Statement 4-7
END Statement 4-7
FINISH Statement 4-8
IIGEN input stream 4-8
Defining the Databases 4-9
MAPGEN Control Statement Group 4-9
Specifying a MAPGEN Group for IMS (DL/I) Databases 4-10
SEGMENT Statements 4-10
FIELD Statements 4-10
Specifying a MAPGEN Group for DB2 Tables and Views 4-11
Specifying a MAPGEN Group for VSAM Non-Hierarchical Data Sets 4-11
Specifying a MAPGEN Group for VSAM Hierarchical Data Sets 4-12
MAPGEN Statement 4-12
SEGMENT Statement for IMS (DL/I) Databases 4-15
RECORD Statement for VSAM Data Sets 4-16
SEGMENT Statement for VSAM Hierarchical Data Sets 4-17
FIELD Statement 4-17
MAPGEN Group Notes 4-25
Using the COBOL Converter 4-26
II.TCUYCNTL (IMSCOBL) 4-27
II.TCUYCNTL (VSAMCOBL) 4-29
COBOL Converter Input Parameters for DL/I 4-30
COBOL Converter Input Parameters for VSAM 4-32
COBOL Features Not Supported 4-35
IMS Considerations 4-35
VSAM Considerations 4-36
User Responsibility 4-36
Using the DB2 Catalog Program 4-36
II.TCUYCNTL (DB2CATL) 4-37
DB2MAP Statement - Input Parameters for the DB2 Catalog Program 4-38
Output from the DB2 Catalog Program 4-39

Samples of Database Definitions 4-41
Sample MAPGEN for a Database 4-41
Sample MAPGEN with a Secondary Index 4-42
Sample DBDGEN with the Same Secondary Index 4-42
Sample MAPGEN with a Variable Length Segment 4-42
Sample DBDGEN with a Variable Length Segment 4-42
Sample VSAM Non-Hierarchical Definition for VSPLANT 4-43
Sample of Defining VSPLANT.ID as an Alternate Key 4-43
Sample of Defining Variable Record in a MAPGEN 4-43
Sample of VSAM Hierarchical Definition for VSHPLANT 4-44
Defining Directories 4-44
DIRECTORY Control Statement Group 4-44
DIRECTORY Statement 4-45
EXCLUDE Statement 4-47
Defining Logical Terminals 4-49
LTERM Control Statement Group 4-49
LTERM Statement 4-49
Defining the Test Data 4-56
IIGEN Sample Input - II.TCUYSRC (IIDMGEN) 4-56
IIGEN Sample Output 4-65
IIGEN Messages 4-65
Maintaining an Application 4-66
UPDATEDIR Statement 4-67
UPDATEDIR Control Statement Group 4-68
DELETE Statement 4-70
DELETE Control Statement Group 4-71

Chapter 5: The Utilities

Introducing the Utilities 5-1
Utility Input Command Statements 5-2
Coding Conventions 5-3
Notation 5-3
IIINIT Utility 5-4
IIGEN Utility 5-4
IXUUNLD Utility 5-4
IXULOAD Utility 5-4
IXUSTAT Utility 5-5
IXUSQRY Utility 5-5
IXUIQRY Utility 5-5
IAOINIT Utility 5-6
IFUINIT Utility 5-6
IFUCLEN Utility 5-6

Accessing the System Database 5-7
Access an IMS (DL/I) System Database 5-7
Access a DB2 System Database 5-8
Accessing a Text Editor Work Database 5-8
Text Editor IMS (DL/I) Work Database 5-8
Text Editor DB2 Work Database 5-9
Accessing the VISION:Journey Download Database 5-9
Initializing the System Database with IIINIT 5-10
INDEX/DIRECTORY Statement 5-10
IIINIT JCL Requirements 5-10
Defining VISION:Inquiry System Database Elements with IIGEN 5-11
Defining Native VISION:Inquiry Messages 5-11
ERRLOAD Control Statement Group 5-12
ERRLOAD Statement 5-12
ERRMSG Statement 5-13
ERRHELP Statement 5-14
MSGLIST Statement 5-16
Defining Vocabularies with IIGEN 5-16
SYSTEM Control Statement Group 5-17
SYSTEM Statement 5-17
SYSLOAD Statement 5-18
Required Commands 5-18
Sample Vocabulary 5-19
IIGEN JCL Requirements 5-19
IIGEN Output 5-20
Maintaining the System Database 5-22
Unloading the System Database with IXUUNLD 5-23
IXUUNLD JCL 5-23
II.TCUYCNTL (IMSUNLD) 5-23
II.TCUYCNTL (DB2UNLD) 5-24
Reloading the System Database with IXULOAD 5-24
OPTIONS Control Statement Group 5-25
OPTIONS Statement 5-25
INCLUDE / EXCLUDE Statements 5-25
END Statement 5-27
Sample OPTIONS Control Statement Group 5-27
IXULOAD JCL Requirements 5-28
II.TCUYCNTL (IMSLOAD) 5-28
II.TCUYCNTL (DB2LOAD) 5-30
Unloading the Stored Inquiries and Functions with IXUSQRY 5-31
IXUSQRY Statement 5-31
IXUSQRY Statement Example: 5-33
IXUSQRY JCL Requirements 5-33
II.TCUYCNTL (IMSSQRY) 5-34

II.TCUYCNTL (DB2SQRY) 5-35
Converting Stored Inquiries and Functions with IXUIQRY 5-36
IXUIQRY JCL Requirements 5-36
II.TCUYCNTL (IMSIQRY) 5-37
II.TCUYCNTL (DB2IQRY) 5-38
IXUIQRY Sample Output 5-38
Reporting System Database Statistics with IXUSTAT 5-39
Understanding the Report Contents 5-39
IXUSTAT Report Contents 5-39
System Database Statistics Report 5-40
Inquiry and Function Report 5-44
Directory Report 5-45
Using the Report Information to Improve Performance 5-46
IXUSTAT JCL Requirements 5-46
II.TCUYCNTL (IMSSTAT) 5-47
II.TCUYCNTL (DB2STAT) 5-47
Initializing the AQF Work Database with IAOINIT 5-48
Initializing the Text Editor Work Database 5-48
Maintaining the Text Editor Work Database with IFUCLEN 5-49
IFUCLEN JCL Requirements 5-49
II.TCUYCNTL (IMSTXTCL) 5-50
II.TCUYCNTL (DB2TXTCL) 5-51
OPTIONS Control Statement Group 5-51
OPTIONS Statement 5-52
END Statement 5-52
Initializing the VISION:Journey Download Database with IFUINIT 5-52
Maintaining the VISION:Journey Download Database with IFUCLEN 5-53
OPTIONS Control Statement Group 5-53
OPTIONS Statement 5-53
END Statement 5-55
OPTIONS Control Statement Group Sample 5-56
IFUCLEN JCL Requirements 5-56
II.TCUYCNTL (IMSFTSCL) 5-57

Chapter 6: Programming and Operation Considerations

Environments and Environment Processors 6-1
Additional Load Modules 6-3
Program Entry Names 6-4
Identification Modules 6-4
System Identification Module, IXSIDENT, and VISION:Inquiry Utilities/Programs 6-5
Building the System Identification Module IXSIDENT 6-6
Specifying the IIIDENT Macro Options for VISION:Journey 6-7

Building the Identification Modules IXBIDENT, IXOIDENT, IXPIDENT, and IXTIDENT 6-7
Applying the Identification module(s) to the system 6-8
Use of Application and Terminal Names 6-10
Conversational Mode 6-10
Continuous Mode 6-11
The II Program for IMS/DC MPP 6-12
Terminal Input Format 6-12
Terminal Output Format 6-13
The IIBMP Program for IMS/DC BMP 6-14
IIBMP Input 6-14
IIBMP Output 6-14
IIBMP DB2 Access 6-14
The IIBATCH Program for Batch 6-15
IIBATCH Input 6-15
IIBATCH Output 6-16
IIBATCH Execution 6-16
IIBATCH Execution JCL to Access IMS (DL/I) System and User Databases 6-16
II.TCUYCNTL (IQBATD and IQBATD2) 6-17
IIBATCH Execution JCL to Access Non-IMS System and User Databases 6-19
II.TCUYCNTL (IQBATV) 6-20
IITSO Program for TSO 6-21
II.TCUYCNTL (DYLIQ) 6-21
The SORT Command 6-25
Allocate Sort Files 6-25
SORT Command Considerations 6-26
Using the EXTRACT Command 6-27
Extract Data Set Format 6-28
Extract Data Set Results 6-29
Extracting Data From Multiple Databases 6-29
EXTRACT Command Considerations 6-30
Text Editor Operation 6-31
Text Editor Screens Considerations 6-31
Text Editor Work Database 6-32
Automatic Query Facility (AQF) Considerations 6-32
AQF Screens (Menus) Consideration 6-32
II.TCUYSRC (AQFIMS) 6-33
Making Modifications to the AQD Screens 6-34
AQF Work Database Size Considerations 6-34
AQF Segment Types 6-35
Intraccess Considerations 6-37
Online Processing 6-37
Programming Considerations 6-37
Using Conversational vs. Continuous Mode with Intraccess 6-38
Downloading Data or a Report with Intraccess 6-38

File Transfer Operation of VISION:Inquiry 6-38
VISION:Journey for Windows Operation 6-39
VISION:Journey Online Design 6-39
VISION:Journey Sequence of Actions 6-40
VISION:Journey Host Considerations 6-40
VISION:Journey Download Database 6-41
II.TCUYSRC (IIDMGEN) 6-42
Description of the Fields 6-43
FTSROOT Segment 6-43
FTSDESC Segment 6-44
FTSRECS Segment 6-44
Native SQL Syntax Facility 6-45
Native SQL Syntax considerations 6-45

Chapter 7: User Exits

Introducing the User Exits 7-1
Programming and Installation Considerations 7-2
Communication Between VISION:Inquiry and User Exits 7-4
Passing Parameters 7-4
Issuing Messages 7-5
Entry Points and Messages 7-5
Assembler Exit Considerations 7-6
IXUSER Macro 7-6
Example Assembler Input Exit 7-8
PL/I Exit Considerations 7-9
Example PL/I Input Exit 7-9
Input Exit IXXUIN 7-10
IXXUIN Parameters 7-11
IXXUIN Parameters Explanations 7-11
Output Exit IXXUOUT 7-12
IXXUOUT for Inquiries with no Native SQL Statement 7-12
Non-UDO Inquiry 7-13
UDO Inquiry 7-13
IXXUOUT Parameters 7-13
IXXUOUT Parameter Explanations 7-14
IXXUOUT Output 7-16
IXXUOUT for Inquiries with Native SQL Statement 7-16
IXXUOUT Parameters 7-17
IXXUOUT Parameter Explanations 7-17
Conversion Exit IXXUCON 7-20
IXXUCON Parameters 7-21
IXXUCON Parameter Explanations 7-21
IXXUCON Considerations 7-22

User Function Exit IXXUFNC 7-25
IXXUFNC Parameters 7-25
IXXUFNC Parameter Explanations 7-26
Invoking IXXUFNC, the Function Exit 7-26
FDVALUE - Function Value Array 7-27
Security Exit IXXUSEC 7-28
IXXUSEC Parameters 7-29
IXXUSEC Parameter Explanations 7-30
VSAM Exit IXXUVSM 7-31
IXXUVSM Parameters 7-31
IXXUVSM Parameter Explanations 7-32

Appendix A: System Vocabulary and Codes

Appendix B: Sample User Exits

Sample User Exits B-1
Sample Conversion Exits (IXXUCONP and IXXUCONS) B-2
IXXUCONP PL/I Sample B-2
IXXUCONS Assembler Sample B-5
Adding the IXHEX Conversion Macro to IXXUCON B-8
IXXUCON Conversion Exit Considerations B-10
Invoking the IXXUCON Conversion Exit B-11
Implementing the Conversion Exit B-13
Sample Function Exit (IXXUFNC) B-14
Source Code for the Sample IXXUFNC Function Exit B-14
Function Exit Table B-17
Sample User Function Error Messages B-18
Installing the Sample Function Exit B-18
IXXUFTM Sample Bit Testing Function B-18
IXXUFTM Bit Testing Function Source B-19
IXXUFNC Function Exit Coding Considerations B-22
Invoking the IXXUFNC Function Exit B-23
Installing the Sample Bit Testing Function IXXUFTM B-24
IXXUFTM Bit Testing Function Error Messages B-24
Sample Input Exit (IXXUIN) B-25
Source Code for the Sample IXXUIN Input Exit B-25
Invoking the IXXUIN Input Exit B-29
Input Exit Security Table (IXXSTABL) B-30
Input Exit Error Messages B-30
Implementing the Sample Input Exit B-31

Appendix C: The IXSECTY and IXSGEN Macros

IXSECTY Macro C-1
IXSGEN Macro C-3

Appendix D: System Modules

Native VISION:Inquiry Modules D-1
Text Editor Modules D-2
AQF Modules D-3
DB2 Modules D-3
VISION:Journey Modules D-4

Appendix E: VISION:Inquiry Target and Distribution Libraries

Source Library E-2
Control Library E-5

Index

Chapter

1

Introducing VISION:Inquiry

Thank you for choosing Advantage™ VISION:Inquiry® 6.5. This product will be 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 IMS\Books\) or a directory of your choice (for example, C:\Advantage VISION_Inquiry 6.5 IMS\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 IMS. 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 eSupport.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 manual.
- 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 as a message processing program (MPP), a batch message processing program (BMP), a DL/I batch program, or an interactive TSO program.
- Operates in MVS in the IMS/DC, BMP, IMS batch, and TSO environments.

  Note: IMS/DC is generic and is generally meant to represent IMS/DC and IMS/TM.

- Utilizes standard coding and design conventions for application programs executing in these environments.
- Supports terminals, display stations, and printers supported by Message Format Service (MFS).

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.
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 an HDAM IMS (DL/I) database. 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	Means DB2 tables or views.
DL/I database	Means the 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 IMS and TSO 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 IMS and TSO 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 IMS and TSO Technical Reference Guide.*
- This document was previously called the *VISION:Inquiry Customer Bulletin*.

Advantage VISION:Inquiry for IMS and TSO Installation Guide

This environment-specific document contains the installation instructions and the Post-Installation Dialog.

Advantage VISION:Inquiry for IMS and TSO 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 IMS and TSO 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.

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:Journeycompact 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.

Chapter

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 IMS 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, 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 VSAM data sets in the BMP region, in batch, and in TSO
- 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 MPP, BMP, batch, and TSO environments
- Checkpoint and non-checkpoint modes (not IMS conversational)
- 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
- Extraction of data from databases to MVS sequential files.

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 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 MFS screens, programs, and a HIDAM (Hierarchical Indexed Direct Access Method) work database. The function and relationship of these components are described in this guide.

The VISION:Inquiry system database is the heart of the system. The utility programs are used to create and maintain the system database. The four inquiry processing programs (inquiry processors) are:

- MPP (Message Processing Program)
- BMP (Batch Message Processing program)
- DL/I batch program
- TSO online executive.

The VISION:Journey host component consists of an HDAM (Hierarchical Direct Access Method) download database, file transfer processing programs, and utility programs.

The VISION:Inquiry System Database

The system database is a root-only IMS (DL/I) HDAM database or a DB2 table. 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
- Logical terminal description
- Conversational scratch pad storage.

The Inquiry Processing Programs

The inquiry processing programs (inquiry processors), enable the end user to create, store, modify, and execute inquiries. The inquiries are checked 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, requests for database information are converted into DL/I calls, SQL calls, and 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 programs. Special considerations and restrictions for each environment are discussed in Chapter 6, "Programming and Operation Considerations".

MPP program

The MPP inquiry processing program supplied as (II) executes as a transaction in an IMS MPP (Message Processing Program) region that can access IMS (DL/I) and DB2 databases. VSAM data sets may not be accessed from the MPP region.

The processing program:

- Displays error messages at the originating terminal.
- Returns data output from the inquiries to the originating terminal, another terminal, or a printer.

BMP program

The BMP inquiry processing program supplied as (IIBMP) executes as a transaction in an IMS BMP (Batch Message Processing) region that can access IMS (DL/I) databases, DB2 tables, and VSAM data sets.

The processing program:

- Displays error messages at the originating terminal.
- Returns data output from the inquiries to the originating terminal, another terminal, or a printer, or a system output data set (SYSOUT).

Batch program

The batch inquiry processing program (IIBATCH) executes as a batch or DL/I batch job step that can access IMS (DL/I) databases, DB2 tables, VSAM data sets, and combinations of these. Input to the batch program is in the form of 80-byte logical records.

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

TSO program

The TSO inquiry processing program (IITSO) executes as an interactive TSO application that can access IMS (DL/I) databases, DB2 tables, and VSAM data sets with no requirement for IMS data communication, while providing all the features of VISION:Inquiry.

A model CLIST is provided which you can customize for your installation.

The processing program:

- Displays error messages at the TSO terminal.
- Returns data output from the inquiries to the terminal, to a data set, or to SYSOUT.

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 MPP Processing

Figure 2-1 on page 2-7 illustrates the interactions of VISION:Inquiry when it executes in the MPP (Message Processing Program) region.

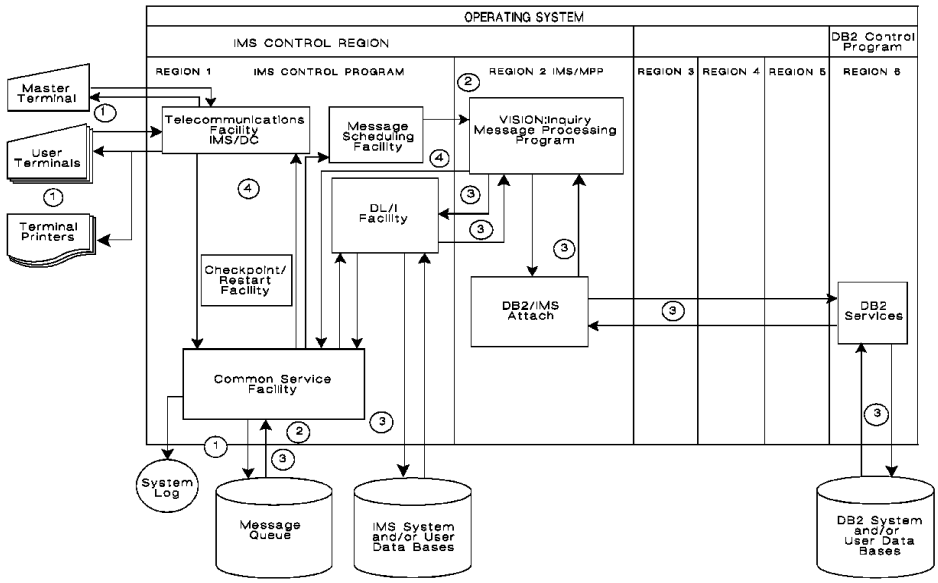


Figure 2-1 The Interactions between VISION:Inquiry and IMS in the MPP Region

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

1. Each inquiry passes from the originating terminal to the IMS control region for scheduling. IMS/DC invokes the Common Service Facility, enters the inquiry on the system log data set, and places it in the message queue.
2. Queued inquiries are scheduled for processing if a message region is available. The scheduling facility receives control and determines, from the transaction code in the inquiry, that VISION:Inquiry is waiting to be scheduled. The VISION:Inquiry MPP modules are scheduled and VISION:Inquiry is loaded into the MPP (Region 2) and given control.
3. VISION:Inquiry retrieves the inquiry as an input message, syntax checks it, then issues calls to the appropriate databases to retrieve the data. The database calls are processed by DL/I or DB2.
4. The output from the processed inquiry is returned to the appropriate terminal, through the message queue.
5. Figure 2-1 shows DL/I as part of the control region, but it does not need to run there. You can run DL/I in its own address space.

VISION:Inquiry BMP Processing

Figure 2-2 illustrates the interactions of VISION:Inquiry when it executes in the BMP (Batch Message Processing) region.

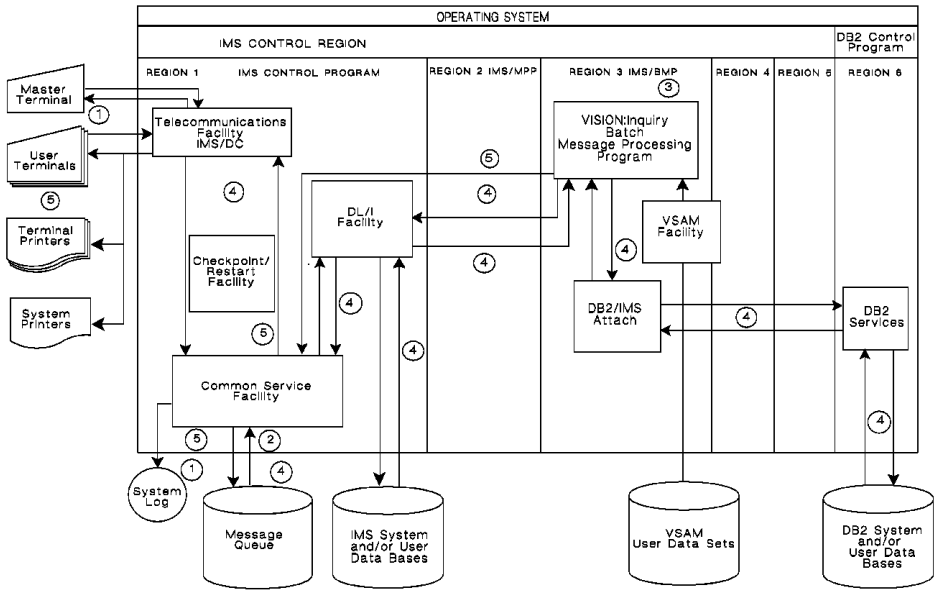


Figure 2-2 The interactions between VISION:Inquiry and IMS in the BMP region

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

1. Each inquiry is submitted from the originating terminal to the IMS message queue for processing. IMS/DC invokes the Common Service Facility, enters the inquiry on the system log data set, and then places it in the message queue.
2. Queued BMP inquiries are scheduled for processing. The Common Service Facilities receive control and determine from the transaction code that BMP inquiries are waiting to be processed.
3. As a result of a computer operator's actions, a batch message processing region (Region 3) is initialized and the VISION:Inquiry BMP program is loaded into the IMS BMP region.
4. VISION:Inquiry retrieves the inquiry as an input message, then issues calls to the appropriate databases to retrieve the data. The database calls are processed by DL/I or DB2. VSAM data sets are accessed using standard VSAM I/O.
5. Output from the processed inquiries is returned to the appropriate terminal or to the system printer. When there are no inquiries in the message queue, the VISION:Inquiry BMP program terminates.

Figure 2-2 shows DL/I as part of the control region, but it does not need to run there. DL/I can be run in its own address space.

VISION:Inquiry Batch Processing

Figure 2-3 illustrates the interactions of VISION:Inquiry when it executes as a DL/I batch processing program. For illustrative purposes, Regions 1 through 3 are omitted from the figure.

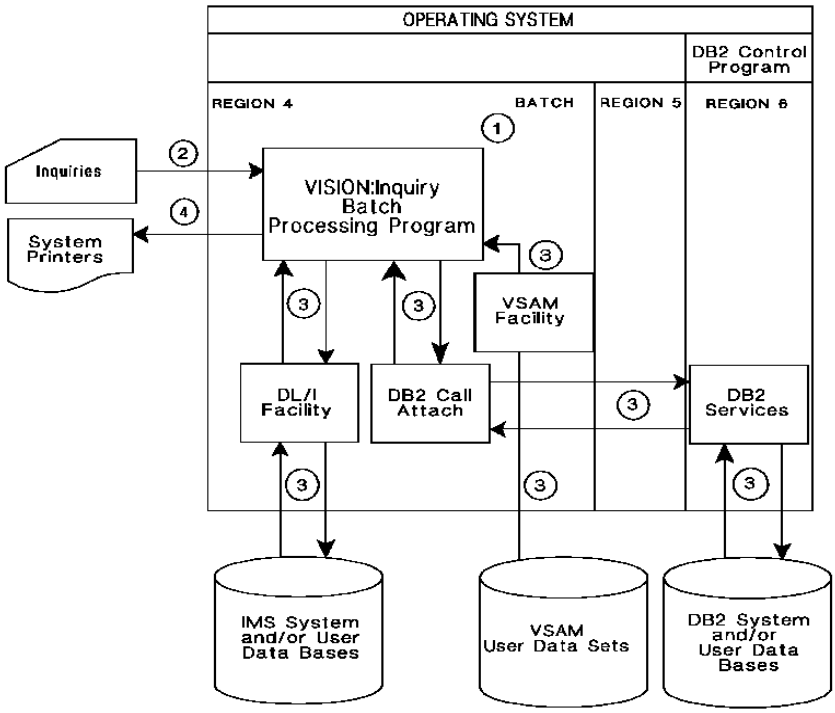


Figure 2-3 VISION:Inquiry in Batch

The following describes the interactions illustrated in Figure 2-3.

1. VISION:Inquiry is invoked in batch Region 4 as the result of a batch job submission. All VISION:Inquiry files are allocated to the job and the appropriate parameters are passed to VISION:Inquiry.
2. Inquiries are supplied in 80-byte records (usually submitted with the JCL for invoking VISION:Inquiry). One or more inquiries can be submitted; each can specify the same or a different logical terminal, and the same or a different transaction code.
3. VISION:Inquiry retrieves inquiries from an input data set, checks syntax, and issues calls to the appropriate databases to retrieve the data. The database

calls are processed by DL/I or DB2. VSAM data sets are accessed using standard VSAM I/O.

4. Output from the processed inquiries is written to the printer. VISION:Inquiry terminates when all the inquiries in the input stream are processed.

VISION:Inquiry TSO Processing

Figure 2-4 illustrates the interactions of VISION:Inquiry when it executes as a TSO program. For illustrative purposes, Regions 1 through 4 are omitted from the figure.

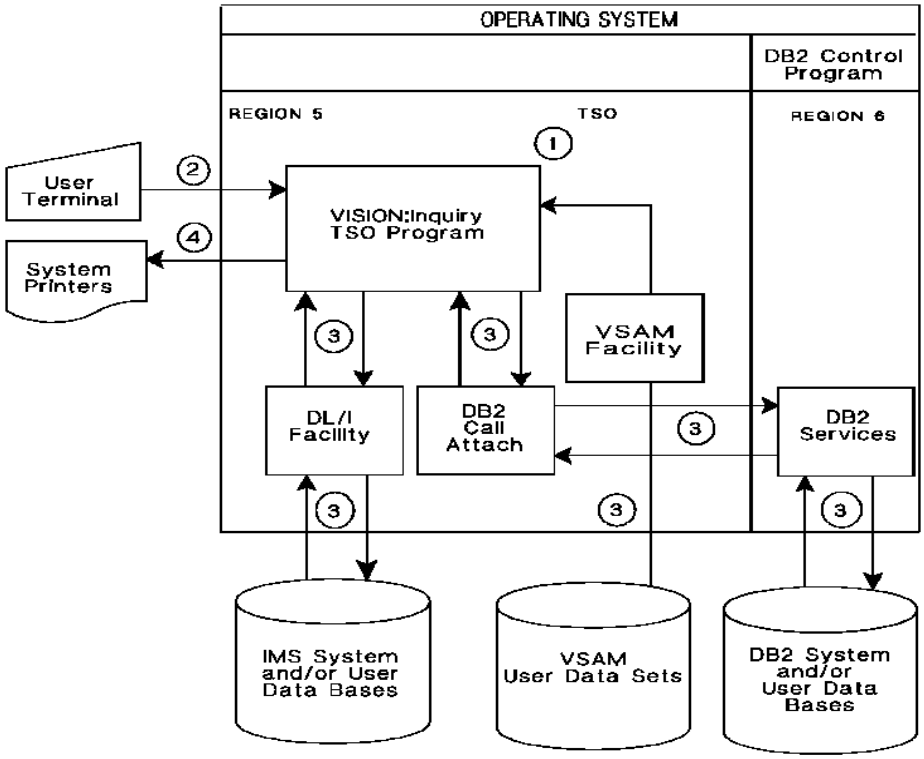


Figure 2-4 VISION:Inquiry in TSO

1. All VISION:Inquiry files are allocated under TSO and VISION:Inquiry is invoked and passed appropriate parameters through the TSO call command.
2. All communications with user terminals are performed by TSO TGET and TPUT macros. Inquiries are submitted from user terminals.
3. VISION:Inquiry retrieves the inquiry from the TSO terminal, checks syntax, then issues calls to the appropriate databases to retrieve the data. The

database calls are processed by DL/I or DB2. VSAM data sets are accessed using standard VSAM I/O.

4. Output from the processed inquiries is returned to the appropriate terminal, or written in the system output data set (SYSOUT) to be subsequently printed.

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 a work database to keep the intermediate results. The work database type is specified at installation time and can be an HDAM IMS (DL/I) database or a DB2 table.

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 and DB2 tables. By selecting options from the menu screens, you can access data fields in the databases and produce reports.

AQF executes in the MPP (Message Processing Program) region and uses a HIDAM IMS (DL/I) work database to keep 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 control information into the download database, and executes an insert to the alternate IOPCB representing the VISION:Journey transaction. This insert operation causes

the VISION:Journey enterprise server component to begin operation within the IMS MPP environment. VISION:Journey communicates with companion software resident in the PC workstation to download the 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 an HDAM IMS (DL/I) download database. For each extraction, the extracted data or report and the necessary information will be written to this download database 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 database.

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 another IMS terminal or printer. To allow VISION:Inquiry to communicate with PC through Intraccess, you need to have the IMS Connect facility installed on your system. See the Intraccess documentation for the 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 the IMS user ID.
VSAM exit	Use to expand or change the VSAM records read from the user file before processing them.

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 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 use the AQF work database
- Create and maintain the Text Editor work database
- 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 or BMP 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.

IAOINIT Utility

The IAOINIT utility initializes the Automatic Query Facility (AQF) work database by creating dummy segments.

IAOINIT executes in the batch environment and is invoked through JCL statements. There are no input parameters or control statements for IAOINIT. Space for AQF work database is allocated through JCL or IDCAMS.

IFUINIT Utility

The IFUINIT utility initializes the VISION:Journey IMS (DL/I) download database or the Text Editor IMS (DL/I) work database which creates a dummy root segment.

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

IFUCLEN Utility

The IFUCLEN utility is the cleanup utility. It deletes the VISION:Journey extracted data or report written in the download database or the Text Editor intermediate results written in the work database. 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 database based on subsequence key, date of extraction, and so on, or the Text Editor intermediate results in the work database.

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.

Chapter

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 or a two-column DB2 table. 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 or a row of a DB2 table.)

Within the 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 IMS and TSO 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.

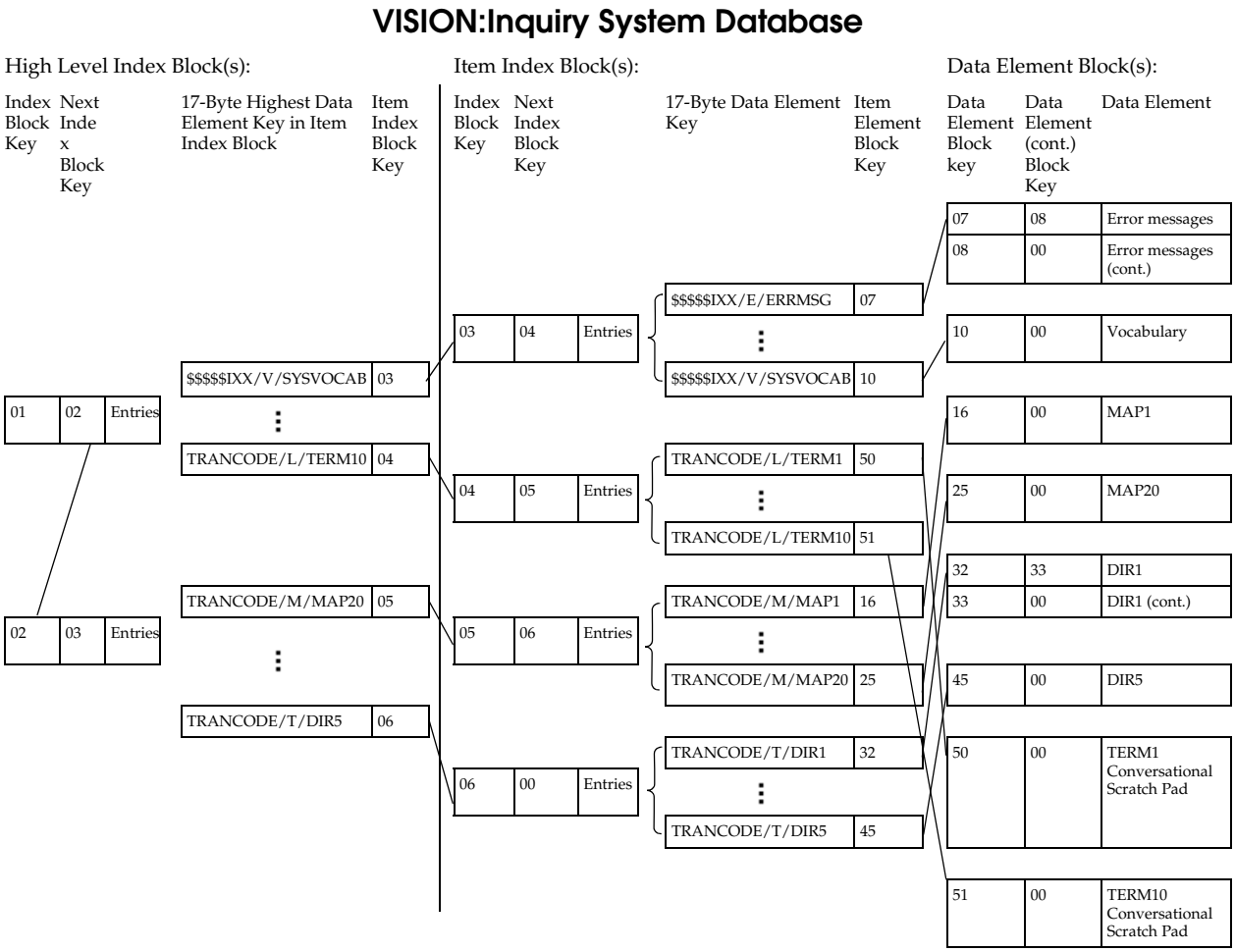


Figure 3-1 VISION:Inquiry System Database

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 IMS and TSO Installation Guide* explain how these may be modified at installation time.

The AQF error messages are defined and hard-coded in a program. The installation procedures in the *Advantage VISION:Inquiry for IMS and TSO Installation Guide* explain how the AQF error messages may be modified at installation time.

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.

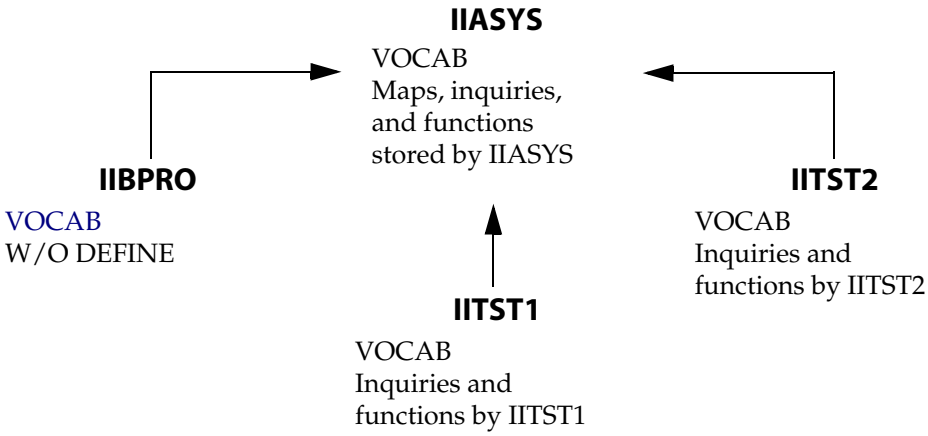


Figure 3-2 A Typical Connected Directory Configuration

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 (LTERMs)

LTERMs and generic LTERMs are the elements in the system database that describe the IMS logical terminals, and generic logical terminals, respectively. The combination of the LTERM or generic LTERM and the IMS transaction code points to the directory; it is the directory that is used to locate the correct database map.

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

VISION:Inquiry logical terminal names must correspond to valid IMS logical terminal names; however, this is not true for generic logical terminal names (see Chapter 4, "The Definition Process"). The logical terminals, which have access to the databases and vocabulary included in a directory, are defined after generation of the directory. Because logical terminals are defined within an application, each logical terminal may be 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 logical terminal.

LTERMs and generic LTERMs 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.

Note: Native VISION:Inquiry is not a conversational system in the IMS/DC sense.

Automatic Query Facility (AQF) Components

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

AQF also uses a three-level HIDAM work database, with six segment types, 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 Download Database

VISION:Journey uses a download database during the transfer of data from the host to a PC file. This download database is implemented as a two level HDAM database. Figure 3-3 shows the structure of the download database.

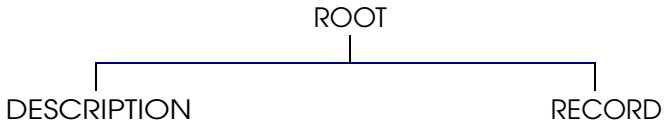


Figure 3-3 VISION:Journey Download Database Structure

- The **root segment** 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 segment** 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 **record segment** contains the actual data that should be downloaded to the PC file.

For a complete description of the fields of the download database segments, see Chapter 6, "Programming and Operation Considerations". There is one more root segment, called the Master Control Record, which is used internally by VISION:Journey. It is also discussed in more detail in Chapter 6.

Text Editor Work Database

The Text Editor uses a work database to save the intermediate results during the editing process. The work database type can be an HDAM database or a DB2 table, and is specified during the installation. The structure of the IMS (DL/I) work database type is the same as the VISION:Journey download database shown in Figure 3-3. It can be shared between the two processing programs (such as the Text Editor and VISION:Journey). Similar information is stored in the DB2 table if the work database type is DB2.

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 3-4 on page 3-12*, DB2 tables (Figure 3-5 on page 3-13), DB2 views (Figure 3-6 on page 3-14), and four VSAM data sets (two of which are pictured in Figure 3-7 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

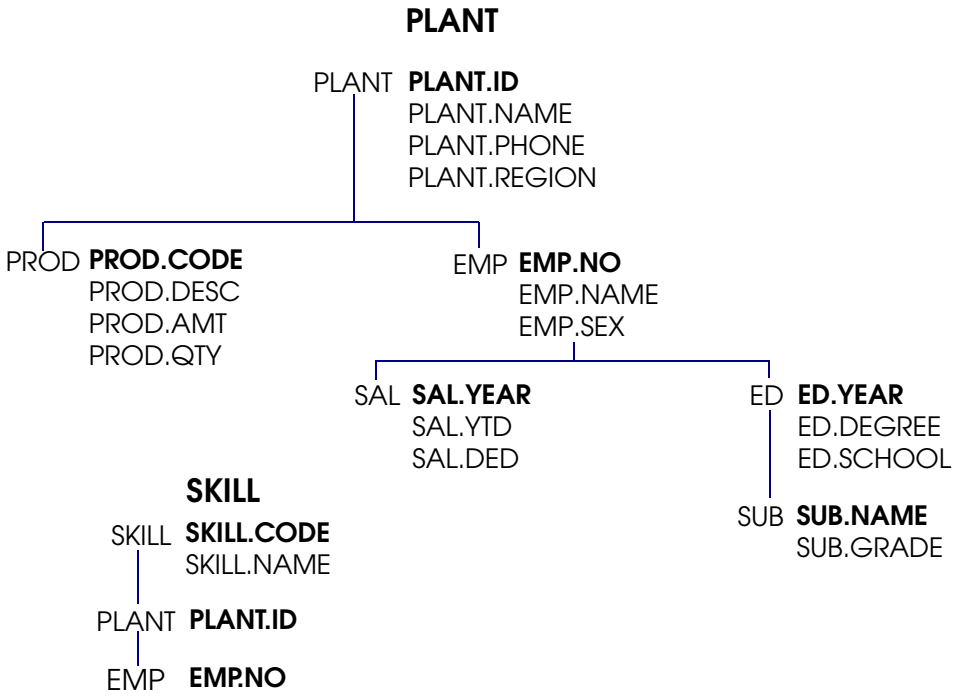


Figure 3-4 IMS (DL/I) Databases, PLANT and SKILL

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

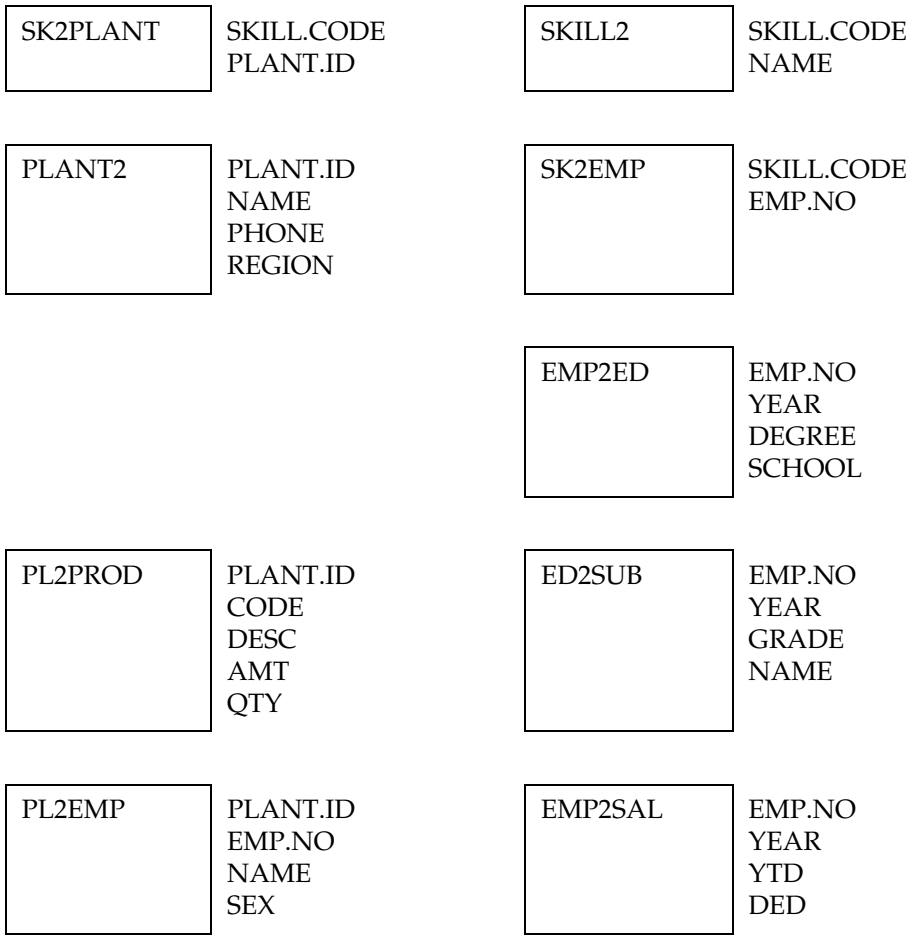


Figure 3-5 DB2 Test Tables

In Figure 3-5, 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 3-6 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 3-7 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 3-8 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 3-8 VSAM Hierarchical Record Layout (In COBOL)

VISION:Inquiry maps the VSAM hierarchical record layout in Figure 3-8 into the hierarchical structure in Figure 3-9. 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.

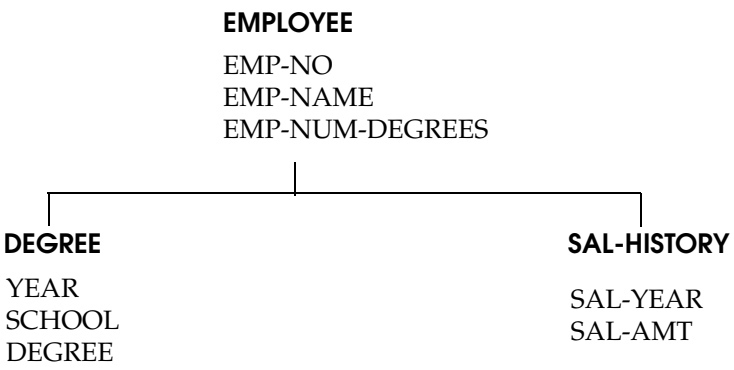


Figure 3-9 VSAM Hierarchical Record Layout (In VISION:Inquiry)

In Figure 3-9, 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.

Chapter

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 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 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 LTERMs. The IIGEN DIRECTORY control statement group defines directories. The following items are specified in the DIRECTORY statement:

- Directory name
- Database map name(s) (not applicable to the native SQL syntax facility)
- Vocabulary name
- BMP transaction code when SORT is deferred
- 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 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 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 logical terminal descriptions (LTERMs).

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 maps, system vocabularies, data fields, or vocabulary words, access to the specified information is restricted.
LTERM	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 LTERMs, a group of LTERMs 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, LTERMs, 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
- LTERM
- SYSTEM
- ERRLOAD
- DELETE

IIGEN Names

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

System Names

All system names used by IIGEN, VISION:Inquiry, and database managers 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 map named M1, a DBD 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 8 character IMS transaction code that identifies the logical application for which IIGEN is to create a VISION:Inquiry environment.

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.TCUYCNTL (IMS ELEM) and II.TCUYCNTL (DB2 ELEM) 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
       LTERM  NAME=LTRM1,DIRECTORY=DIR1,....
       FINISH
       LTERM  NAME=LTRM*,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 BMP or batch region JCL or the TSO ALLOCATE command before the inquiry program is run.

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",)

NAME=name(,NAME=name)...
```

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",)

NAME=name(,NAME=name)...
```

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,)

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

NAME=name(,NAME=name)...
```

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.TCUYCNTL (IMSCOBL)

The control library member II.TCUYCNTL (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=II
*
*    COBOL DEFINITION EXAMPLE FOR PLANT
*
MAPGEN MAP=IIDMMAP,DBD=IIDBDDM,NAME=PLANT
 SEGMENT SEGM=PLANT,COPYCOBOL=COBPLANT,PSB=II
 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=II
 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.TCUYCNTL (VSAMCOBL)

The control library member II.TCUYCNTL (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=II
*
*    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, LTERM 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=II
```

  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=II
$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=II
  ```

  For CA-Librarian:

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

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

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

  or

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

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, LTERM 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 4-35, and VSAM Considerations on page 4-36, 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.TCUYCNTL (DB2CATL)

The control library member II.TCUYCNTL (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.DSNLOAD,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,
       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,
```

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

```
       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
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.
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
 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

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.

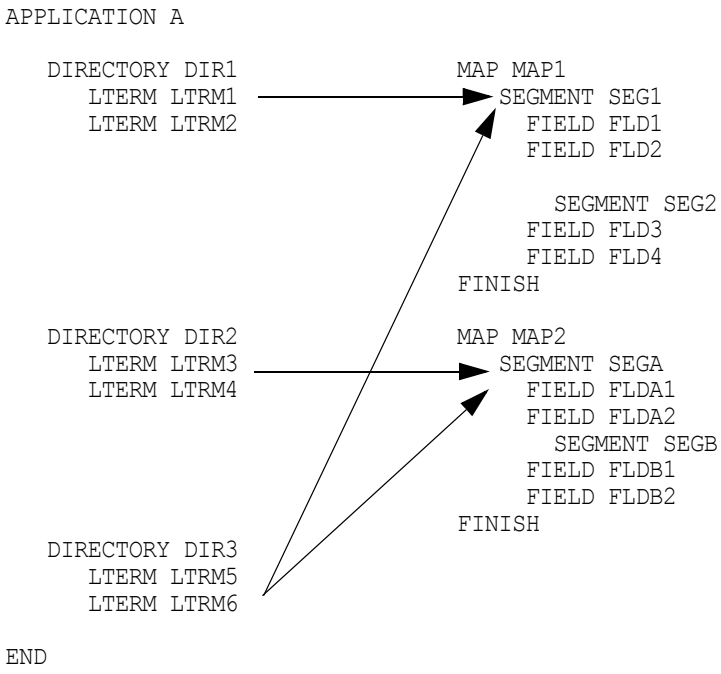


Figure 4-14 A Typical Application's Directory Configuration

DIRECTORY Statement

The format of the DIRECTORY statement is:

DIRECTORY NAME=name(,MAP=mapname)(,VOCAB=vocabname)
(,STRAN=tran-name)(,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 LTERM 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.

STRAN= tran-name

Specifies the BMP transaction code used for the message switch when SORT is deferred to the BMP. See the LTERM Statement on page 4-49 for how to defer SORT.

SLIMIT=num

Specifies the maximum number of database calls allowed when SORT is specified by an MPP inquiry.

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

If the number of calls 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 Logical Terminals

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

LTERM Control Statement Group

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

Each LTERM 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 LTERM statements which point to the same directory can be specified.

LTERM Statement

The LTERM statement contains the following information:

- Logical terminal name or generic terminal name
- Directory name to which the LTERM is related
- Physical characteristics of the terminal (line width, page height)
- MFS name for IMS terminals
- Limits for database calls and logical pages
- Terminal mode setting
- Whether or not it is a dummy terminal

The format of the LTERM statement is:

```
LTERM     NAME={term-name | generic-term-name},
or        DIRECTORY=directory-name,
TERM      MODE=cvs
          (,TERMLTH=number)
          (,TIME=time)
          (,PAGE=num)
          (,MFS=mfs-name)
          (,TERMWDTH=width)
          (,ERRHELP=val)
          (,CKPT=ckpt)
          (,DEFINITION=lterm)
          (INSERTSIZE=lines)
          (,MFSFLDLEN=characters)
          (,APPL=application)
          (,PCPAGE=pcnum)
          (,PCOUT=YES)
```

The LTERM statement parameters are:

NAME=term-name

Defines the primary logical terminal name for an IMS terminal in the IMS LTERM list. IMS alias names cannot be used since they cause VISION:Inquiry to receive a'Not Found' condition.

- When using VISION:Inquiry in TSO or batch mode, any LTERM can be specified.
- When using TSO or batch mode, enter a /IAM LTERM statement.

NAME= generic-term-name

Defines actual characters and wildcard (*) characters. As an example, the generic term name, LT*0*1*, matches any seven-character terminal name that starts with LT followed by any character, a zero, any character, a 1, and finally any character or blank. A generic terminal name can also be all asterisks.

DIRECTORY= directory-name

Defines the name of the directory with which this LTERM 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=100 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 MPP and TSO environments.

v = 0

Specifies a non-MFS terminal.

v = 1

Specifies an MFS terminal.

s = 0

Specifies that sorts are to be deferred and a message switch to the STRAN transaction code is to be done.

- If sorts are to be deferred, but no STRAN transaction code is specified on the DIRECTORY statement, the inquiry is read and translated, and any syntax errors are reported.
- If there are no syntax errors, the inquiry is temporarily deferred and rescheduled by a message switch to the same transaction code.

VISION:Inquiry then sorts in the MPP. The message processing region is then freed between the translation and the execution of the inquiry.

s = 1

Specifies that the sort is to be done online.

The sort option of the mode parameter can also be applied to the "ORDER BY" command of the native SQL syntax facility.

TERMLTH=number

Specifies the number of lines of available output data per logical page (screen). This parameter is required because of the differences in physical characteristics among terminals. Its specification ensures that proper paging is used for the inquiry output.

- If the TERMLTH parameter is not used, a default of 20 lines is assumed.
- This value should be consistent with the number of output lines in your MFS.
- 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 LTERM statement parameters.

MFS=mfs-name

Specifies the name of a redefined MFS description if this terminal is a video display used by VISION:Inquiry in the MPP or BMP regions. It should also be specified if a particular MFS format is needed. The default when an output message is sent is the input MFS format name.

TERMWDTH=width

Specifies the number of characters that appears in each line for the named logical terminal. 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=lterm

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

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

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

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

INSERTSIZE=lines

Defines a 2-character value that represents the number of lines that are inserted in the message queue with one INSERT call. If INSERTSIZE is not specified, it defaults to 20.

MFSFLDLEN= characters

Specifies the length of the MELD statements of the MFSGEN used for this terminal. If MFSFLDLEN is not specified, it defaults to 119.

APPL=application

Used in conjunction with the DEFINITION parameter. APPL refers back to an LTERM 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.

This parameter is only used with the PCE command. See the notes below regarding LTERM 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 MPP or TSO environments.

- An inquiry that browses through a large database can tie up the 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 MPP region. The first limit is a number of database calls. 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 MPP region is freed 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 MPP region usage.
- In continuous mode, with the exception of batch, the database call 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 online (MPP) 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, MFS, MFSFLDLEN, INSERTSIZE, 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 MFS, MFSFLDLEN, INSERTSIZE, 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, MFS, TERMWDTH, MFSFLDLEN, INSERTSIZE, 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 MFS, MFSFLDLEN, INSERTSIZE, 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 LTERM 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.TCUYSRC (IIDMGEN) which can be used as input to IIGEN.

IIGEN Sample Input - II.TCUYSRC (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=II
*
*        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"
```

Figure 4-16 Input to IIGEN (Page 1 of 9)

```
 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,
         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,
```

Figure 4-16 Input to IIGEN (Page 2 of 9)

```
          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,
          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,
```

Figure 4-16 Input to IIGEN (Page 3 of 9)

```
         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"
*
 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"
```

Figure 4-16 Input to IIGEN (Page 4 of 9)

```
 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"
 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
```

Figure 4-16 Input to IIGEN (Page 5 of 9)

```
*
 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"
 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
```

Figure 4-16 Input to IIGEN (Page 6 of 9)

```
*
 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,
         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"
```

Figure 4-16 Input to IIGEN (Page 7 of 9)

```
 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
 FINISH
*
*        DIRECTORY FOR
```

Figure 4-16 Input to IIGEN (Page 8 of 9)

```
*        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=110,TIME=10,PAGE=2
 FINISH
 TERM   NAME=TSO,DIRECTORY=IIDMDIR,TERMWDTH=79,TERMLTH=20,MODE=100
 FINISH
 TERM   NAME=UDO,DIRECTORY=IIDMDIR,TERMWDTH=79,TERMLTH=12,MODE=110,
        TIME=10,PAGE=2,MFS=INQUDO
 FINISH
 END
```

Figure 4-16 Input to IIGEN (Page 9 of 9)

The IIGEN input, II.TCUYSRC (IIDMGEN), also contains the MAPGEN for the VISION:Journey download database 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 IMS 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 'TSO     ' HAS BEEN INSERTED FOR
IXG0058  250 00  THE DIRECTORY NAMED 'IIDMDIR '
IXG0057  252 00  A DIRECTORY LTERM/TERM NAME 'UDO     ' HAS BEEN INSERTED FOR
IXG0058  252 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 LTERM/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. LTERM

   If an LTERM needs to be modified, only that LTERM 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 LTERM 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)(,STRAN=tran-name)
```

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. I
- f 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, SLIMIT, and STRAN 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 4-44.

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 4-44, 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 4-6, 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=II
UPDATEDIR  NAME=IIDMDIR,VOCAB=VOCAB1,SLIMIT=1000
FINISH
END
```

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

Example 2:

```
APPL  NAME=II
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 II. It also excludes the commands OUTPUT and EXTRACT from the replaced vocabulary.

Example 3:

```
APPL  NAME=II
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 II. 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
- LTERMs
- Vocabularies
- All the elements of an application

The format of the DELETE statement is:

DELETE DIRECTORY=dir-name$_a$)(,LTERM=lterm-name$_b$)(,MAP=map-name$_c$)(,VOCAB=vocab-name$_d$)(,ALL)

The DELETE statement parameters are:

DIRECTORY= dir-name$_a$	Represents a valid directory name to be deleted from the application. Additionally, all LTERMs or generic LTERMs associated with this directory are also deleted. Any connected directory to the deleted directory must be re-entered in the system database.
LTERM= lterm-name$_b$	Specifies the name of an LTERM to be deleted.
MAP= map-name$_c$	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$_d$	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, LTERMs, 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 4-6.

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=II
DELETE DIRECTORY=PAY1,MAP=MAPA
FINISH
END
```

Chapter

5

The Utilities

This chapter describes the VISION:Inquiry utility programs for creating and maintaining the system database, the Text Editor work database, AQF work database, and the VISION:Journey download database .

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 LTERMs.
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.
IAOINIT	Initializes the AQF work database.
IFUINIT	Initializes the Text Editor IMS (DL/I) work database and the VISION:Journey download database.
IFUCLEN	Deletes the information left from previous editing processes from the Text Editor work database. Deletes unwanted extracted data from the VISION:Journey download database.

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
IAOINIT	None
IFUINIT	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.

The IIINIT utility uses the INDEX/DIRECTORY statement as input. The two parameters specify the number of database segments and index segments to be formatted. The segments initialized by IIINIT are the only ones used by VISION:Inquiry; no new segments are added and no old segments 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 LTERMs) 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.TCUYCNTL library members IQBATD (for an IMS system database) or IQBATV (for a DB2 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.TCUYCNTL library members IQBATD (for an IMS system database) or IQBATV (for a DB2 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.

IAOINIT Utility

The IAOINIT utility initializes the AQF work database by creating dummy segments.

There are no input statements for this utility.

IFUINIT Utility

The IFUINIT utility initializes the Text Editor IMS (DL/I) work database and the VISION:Journey download database. It creates a dummy root segment with keys of all zeroes. (The control library member II.TCUYCNTL (DB2TXTIN) creates and initializes the Text Editor DB2 work database.)

There are no input statements for this utility.

IFUCLEN Utility

The IFUCLEN utility is used for two purposes:

- To delete the information from previous editing processes from the Text Editor work database.
- To delete the unwanted data extracted and written to the VISION:Journey download database.

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) or a DB2 system database.

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

Utility	II.TCUYCNTL members for an IMS (DL/I) system database	II.TCUYCNTL members for a DB2 system database
IIINIT	IMSINIT	DB2INIT
IIGEN	IMSELEM	DB2ELEM
IXUUNLD IXULOAD	IMSUNLD IMSLOAD	DB2UNLD DB2LOAD
IXUIQRY IXUSQRY	IMSIQRY IMSSQRY	DB2IQRY DB2SQRY
IXUSTAT	IMSSTAT	DB2STAT

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.TCUYSRC (IIPSB01) contains an example of the load PSB required by IIINIT.
- II.TCUYSRC (IIPSB02) is appropriate for the other utilities.

Whenever DLIBATCH (or, for IIGEN, IMSBATCH) is used, the first database PCB in the PSB represents the system database. Be sure 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 STEPLIB or the link list (such as by including DSN510.DSNLOAD in the STEPLIB concatenation) for successful access to DB2 data.

Accessing a Text Editor Work Database

The Text Editor work database can be either an IMS (DL/I) or DB2 database.

Text Editor IMS (DL/I) Work Database

For a Text Editor IMS (DL/I) work database, the following are used:

- Model JCL to execute the IFUINIT utility can be found in the control library member II.TCUYCNTL (IMSTXTIN).
- Model JCL to execute the IFUCLEN utility can be found in the control library member II.TCUYCNTL (IMSTXTCL).
- The IFUINIT and IFUCLEN utility programs are run using DLIBATCH procedure.
- The name of a PSB containing a PCB for the Text Editor IMS (DL/I) work database is specified as a parameter to the procedure.
  - II.TCUYSRC (FTSPSBL) contains an example of the PSB required by IFUINIT.
  - II.TCUYSRC (FTSPSBC) contains an example of the PSB appropriate for IFUCLEN.

Whenever DLIBATCH is used, the first database PCB in the PSB represents the Text Editor IMS (DL/I) work database. Be sure that the corresponding DD statement references the work database you wish to use.

The same IMS (DL/I) database can also be used as the VISION:Journey download database.

Text Editor DB2 Work Database

For a Text Editor DB2 work database, the following are used:

- Model JCL to allocate and initialize the work database can be found in the control library member II.TCUYCNTL (DB2TXTIN).
- Model JCL to execute the IFUCLEN utility can be found in the control library member II.TCUYCNTL (DB2TXTCL).
- The IFUINIT and IFUCLEN utility programs are run in an ordinary batch region.

Accessing the VISION:Journey Download Database

For the VISION:Journey download database, the following are used:

- Model JCL to execute the IFUINIT utility can be found in the control library member II.TCUYCNTL (IMSFTSIN).
- Model JCL to execute the IFUCLEN utility can be found in the control library member II.TCUYCNTL (IMSFTSCL).
- The IFUINIT and IFUCLEN utility programs are run using the DLIBATCH procedure.
- The name of a PSB containing a PCB for the VISION:Journey download database is specified as a parameter to the procedure.
  - II.SCRLIB (FTSPSBL) contains an example of the PSB required by IFUINIT.
  - II.TCUYSRC (FTSPSBC) contains an example of the PSB appropriate for IFUCLEN.

Whenever DLIBATCH is used, the first database PCB in the PSB represents the VISION:Journey download database. Be sure that the corresponding DD statement references the download database you wish to use.

The same database can also be used as the Text Editor IMS (DL/I) work database.

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.

INDEX/DIRECTORY Statement

The number of segments 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.TCUYCNTL (IMSINIT). A listing of this member is given in the Defining and Initializing the System Database section of the *Advantage VISION:Inquiry for IMS and TSO 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.TCUYCNTL (DB2INIT). A listing of this member is given in the Defining and Initializing the System Database section of the *Advantage VISION:Inquiry for IMS and TSO Installation Guide*.

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 LTERMs) 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.TCUYSRC (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.TCUYSRC (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 with them.

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.TCUYSRC (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
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

IIGEN JCL Requirements

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

Sample JCL to execute IIGEN using a DB2 system database is found in control library member II.TCUYCNTL (DB2ELEM). For a listing see the Defining a Test Application section of the *Advantage VISION:Inquiry for IMS and TSO 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 5-20 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 VISION:Inquiry for IMS, CICS, and TSO Error and Informational Messages manual.

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 IMS 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
```

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

```
                 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
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 IMS SYSTEM DEFINITION AND GENERATION DIAGNOSTICS
```

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

```
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 3 of 3)

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 LTERM 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.TCUYCNTL (IMSUNLD)
- II.TCUYCNTL (DB2UNLD)

II.TCUYCNTL (IMSUNLD)

Control library member II.TCUYCNTL (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.
//*
//*
//IXUUNLD EXEC DLIBATCH,MBR=IXUUNLD,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//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.TCUYCNTL (DB2UNLD)

Control library member II.TCUYCNTL (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.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//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

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 or DB2) 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 4-6.) 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  }  (APPL=applname)(,DIRECTORY=dirname)
{ INCLUDE  }  (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
```

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.

Statement	Explanation <$paranum
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.TCUYCNTL (IMSLOAD)
- II.TCUYCNTL (DB2LOAD)

II.TCUYCNTL (IMSLOAD)

Control library member II.TCUYCNTL (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) )
```

Figure 5-6 Sample JCL to Reload an IMS (DL/I) System Database (Page 1 of 2)

```
/*
//*
//*  INITIALIZE THE SYSTEM DATABASE
//*
//IIINIT   EXEC DLIBATCH,MBR=IIINIT,PSB=IIPSB01
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//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.TCUYPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//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 5-6 Sample JCL to Reload an IMS (DL/I) System Database (Page 2 of 2)

II.TCUYCNTL (DB2LOAD)

Control library member II.TCUYCNTL (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.TCUYCNTL 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.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//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.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=II.IXXDB.UNLOAD
//SYSIN    DD *
END
/*
```

Figure 5-7 Sample JCL to Reload a DB2 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.TCUYCNTL (IMSSQRY)
- II.TCUYCNTL (DB2SQRY)

II.TCUYCNTL (IMSSQRY)

Control library member II.TCUYCNTL (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.
//*   - ADD THE NECESSARY INPUT CONTROL STATEMENTS AS SYSIN DATA.
//*
//*
//IXUSQRY  EXEC DLIBATCH,MBR=IXUSQRY,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//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 5-8 Sample JCL to Unload Stored Inquiries or Functions from an IMS (DL/I) System Database

II.TCUYCNTL (DB2SQRY)

Control library member II.TCUYCNTL (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.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//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 5-9 Sample JCL to Unload Stored Inquiries/Functions from a DB2 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.TCUYCNTL (IMSIQRY)
- II.TCUYCNTL (DB2IQRY)

II.TCUYCNTL (IMSIQRY)

Control library member II.TCUYCNTL (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.
//*   - 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.TCUYPGM
//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 5-10 Sample JCL to Convert Inquiries/Functions for an IMS (DL/I) System Database

II.TCUYCNTL (DB2IQRY)

Control library member II.TCUYCNTL (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.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//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 5-11 Sample JCL to Convert Inquiries/Functions for a DB2 System Database

IXUIQRY Sample Output

Figure 5-12 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 5-12 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 5-13 on page 5-40. 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

The following figure 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	II					
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		

Figure 5-13 Sample System Database Statistics Report (Page 1 of 2)

```
    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
  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            II           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 5-13 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 LTERMs, that have been permitted access to VISION:Inquiry because they match a generic LTERM.
- At (19), these statistics are the total of entries, bytes, average bytes, and directory blocks used for inquires deferred from a LTERM that gained access to VISION:Inquiry because it matched a generic LTERM.

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 LTERM names stored for this generic LTERM (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. The following figure shows a sample Inquiry and Function Report.

```
VISION:Inquiry 6.5 FOR IMS          STORED INQUIRIES AND FUNCTIONS                                PAGE 1

                                        ————————CREATED————————  ————————  MODIFIED  ————————INT     SRC
TYPE INQUIRY-OR-FUNCTION-NAME-APPL--DIRECT-TERMINAL-DATE  ———  TIME—— TERMINAL-DATE———    TIME—BYTES — BYTES

INQY BONUS.PLUS               II    ALL    BATCH    2002-05-28 14:46  DY3      2002-05-30 16:33  200     350
INQY BONUS.SKILL              II    ALL    BATCH    2002-05-28 14:46  ******** ******   ****     748     500
INQY CONV.EXIT                II    ALL    BATCH    2002-05-28 14:46  DY3      2002-05-30 16:35  200     220
INQY EMPLOYEE.LISTING         II    ALL    BATCH    2002-05-28 14:46  *******  ******   ****     296     190
INQY EMPLOYEE.SALARY          II    ALL    BATCH    2002-05-28 14:46  *******  ******   ****     928     462
FUNC MON.SAL                  II    ALL    BATCH    2002-05-28 14:46  *******  ******   ****     244     201
INQY SAL.DATA                 II    ALL    BATCH    2002-05-28 14:46  *******  ******   ****     244     484
UNC YEARLY.BONUS              II    ALL    BATCH    2002-05-28 14:46  DY2      2002-06-02 11:11  220     284
```

Figure 5-14 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.

The following figure shows a sample directory report.

```
VISION:Inquiry 6.5 FOR IMS                              D I R E C T O R Y   S T A T I S T I C S          PAGE   1
COMPONENT------------- ENTRY-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME  MAP-NAME

APPL.................   II
  DIRECTORY.........   ALL
    MAPS..............             IIDMMAP   IIDSMAP   FTSMAP
  DIRECTORY.........   IIDMDIR
    MAPS..............             **** NONE FOUND.
  DIRECTORY.........   VSAM
    MAPS..............             VSPLMAP  VSSKMAP   VSHPLMAP  VSSKMAP

VISION:Inquiry 6.5 FOR IMS                                D I R E C T O R Y   S T A T I S T I C S          PAGE   2

     PRIMARY                                    CONNECTED
APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--      APPL----/DIRECT--

II      /ALL

II      /IIDMDIR  ===> II      /ALL

II      /VSAM     ===> II      /IIDMDIR  ===> II      /ALL
```

Figure 5-15 Sample Directory Report

Page 1 of the report in Figure 5-15 shows that the system database contains one application (II) 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.TCUYCNTL (IMSSTAT)
- II.TCUYCNTL (DB2STAT)

II.TCUYCNTL (IMSSTAT)

Control library member II.TCUYCNTL (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.
//*
//*
//IXUSTAT  EXEC DLIBATCH,MBR=IXUSTAT,PSB=IIPSB02
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//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 5-16 Sample JCL for Generating Reports from the IMS (DL/I) System Database

II.TCUYCNTL (DB2STAT)

Control library member II.TCUYCNTL (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.TCUYPGM
//          DD DISP=SHR,DSN=DSN510.DSNLOAD
//SYSPRINT  DD SYSOUT=*
//DRPRINT   DD SYSOUT=*
//IFPRINT   DD SYSOUT=*
```

Figure 5-17 Sample JCL for Generating Reports from the DB2 System Database

Initializing the AQF Work Database with IAOINIT

The IAOINIT utility initializes the AQF work database. The model JCL for executing the AQF utility can be found in control library member II.TCUYCNTL (AQFINIT). A listing for this member is given in the Defining and Initializing the AQF Work Database section of the *Advantage VISION:Inquiry for IMS and TSO Installation Guide*.

The ddname used for the AQF work database must match that specified in the DBD.

This utility program is run using the DLIBATCH procedure. The name of a PSB containing a PCB of the AQF work database is specified as a parameter to the procedure.

The source library member II.TCUYSRC (AQFPSBIN) contains an example of the PSB required by IAOINIT.

Whenever DLIBATCH is used, the first database PCB in the PSB represents the AQF work database. Be sure that the corresponding DD statement references the work database you wish to use.

Initializing the Text Editor Work Database

The Text Editor work database is initialized by creating a dummy segment/record.

- The IFUINIT utility creates the dummy segment if the work database type is IMS.
- The SQL statement, INSERT, is used to create the dummy record if the work database type is DB2.

The control library member II.TCUYCNTL (IMSTXTIN) contains the sample JCL to allocate and initialize the Text Editor IMS (DL/I) work database. A listing of this member is given in the Defining and Initializing the Text Editor Work Database section of the *Advantage VISION:Inquiry for IMS and TSO Installation Guide*. The ddname used for the work database must match that specified in DBD.

The control library member II.TCUYCNTL (DB2TXTIN) contains the sample JCL to allocate and initialize the Text Editor DB2 work database. A listing of this member is given in the Defining and Initializing the Text Editor Work Database section of the *Advantage VISION:Inquiry for IMS and TSO Installation Guide*.

Maintaining the Text Editor Work Database with IFUCLEN

The IFUCLEN utility is used to delete the information remaining in the Text Editor work database from previous editing process based on the parameters specified in the OPTIONS control statement.

IFUCLEN JCL Requirements

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

- II.TCUYCNTL (IMSTXTCL)
- II.TCUYCNTL (DB2TXTCL)

II.TCUYCNTL (IMSTXTCL)

If the work database type is IMS, the utility runs as a batch job using DLIBATCH procedure and uses the first database PCB in the PSB as the work database. The ddname used for the work database must match that specified in DBD and the data set name should reference the one that you wish to use.

The control library member II.TCUYCNTL (IMSTXTCL) contains the sample JCL for maintaining the Text Editor IMS (DL/I) work database.

```
//* TEXT EDITOR IMS WORK DATABASE CLEANUP JCL:
//*   - THIS JOB WILL DELETE FROM THE WORK DATABASE THE USER
//*     INFORMATION STORED FROM THE LAST TEXT EDITOR PROCESSING.
//*     THE PARAMETERS IN OPTIONS CONTROL STATEMENT SPECIFY
//*     THE INFORMATION OF WHICH USER(S) SHOULD BE DELETED.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS REQUIRED.
//*   - VERIFY OR CHANGE THE WORK DATABASE DD NAMES AS REQUIRED.
//*   - CHANGE AND VERIFY THE PARAMETERS OF THE OPTIONS CONTROL
//*     STATEMENT TO REFLECT THE DATA WHICH SHOULD BE DELETED.
//*   - THE DBDS AND PSBS MUST ALREADY HAVE BEEN GENERATED.
//*
//*
//*  CLEANUP THE IMS WORK DATABASE
//*
//IFUCLEN  EXEC DLIBATCH,MBR=IFUCLEN,PSB=FTSPSBC,RGN=2048K
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//FTSROOT  DD DSN=II.IDXFTS.ROOT,DISP=SHR
//FTSDESC  DD DSN=II.IDXFTS.DESC,DISP=SHR
//FTSRECS  DD DSN=II.IDXFTS.RECS,DISP=SHR
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
8192,6
//FTSDEL   DD SYSOUT=*
//SYSIN    DD *
 OPTIONS APPL=XXXXXXXX,LTERM=XXXXXXXX,TXT-EDIT=Y
 END
/*
```

Figure 5-18 Sample JCL to Maintain the Text Editor IMS (DL/I) Work Database

II.TCUYCNTL (DB2TXTCL)

If the Text Editor work database type is DB2, the utility runs in an ordinary batch region. The control library member II.TCUYCNTL (DB2TXTCL) contains the sample JCL for maintaining the Text Editor DB2 work database.

```
//* TEXT EDITOR DB2 WORK DATABASE CLEANUP JCL:
//*   - THIS JOB WILL DELETE FROM THE WORK DATABASE THE USER
//*     INFORMATION STORED FROM THE LAST TEXT EDITOR PROCESSING
//*     IF THE DATABASE TYPE IS DB2.
//*     THE PARAMETERS IN OPTIONS CONTROL STATEMENT SPECIFY
//*     THE INFORMATION OF WHICH USER(S) SHOULD BE DELETED.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS REQUIRED.
//*   - CHANGE AND VERIFY THE PARAMETERS OF THE OPTIONS CONTROL
//*     STATEMENT TO REFLECT THE DATA WHICH SHOULD BE DELETED.
//*
//*
//*  CLEANUP THE DB2 WORK DATABASE
//*
//IFUCLEN  EXEC PGM=IFUCLEN,REGION=2M
//STEPLIB  DD DISP=SHR,DSN=II.TCUYPGM
//         DD DISP=SHR,DSN=DSN510.DSNLOAD
//SYSPRINT DD SYSOUT=*
//FTSDEL   DD SYSOUT=*
//SYSIN    DD *
 OPTIONS APPL=XXXXXXXX,LTERM=XXXXXXXX,TXT-EDIT=Y
 END
/*
```

Figure 5-19 Sample JCL to Maintain the Text Editor DB2 Work Database

The utility also creates a report listing of all the records deleted.

OPTIONS Control Statement Group

By specifying the parameters in the OPTIONS control statement, you can control one or more sets of information to be deleted from the Text Editor work database.

You can have as many OPTIONS statements in an OPTIONS control statement group as you need:

```
OPTIONS statement
        .
        .
        .
OPTIONS statement
END
```

OPTIONS Statement

The format of the OPTIONS statement is:

OPTIONS TXT-EDIT=Y (,APPL=applname)(,LTERM=termname)

The OPTIONS statement parameters are:

TXT-EDIT=Y	Specifies the Text Editor information to be deleted. This parameter must be specified.
APPL= applname	Specifies the logical application name to which the deletion applies. If omitted, no application checking is done for the deletion.
LTERM= termname	Specifies the terminal name to which the deletion applies. If omitted, no terminal name checking is done for deletion.

At least one of the parameters, APPL or LTERM, must be specified.

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 required.

Initializing the VISION:Journey Download Database with IFUINIT

The VISION:Journey download database is initialized by creating a dummy root segment. The IFUINIT utility creates the dummy root segment and initializes the subsequence number to 0.

The control library member II.TCUYCNTL (IMSFTSIN) contains the sample JCL for initializing the VISION:Journey download database. You can find a listing of this member in the Defining and Initializing the VISION:Journey Download

Database section of the *Advantage VISION:Inquiry for IMS and TSO Installation Guide*. The ddname used for the VISION:Journey download database must match that specified in the DBD.

Maintaining the VISION:Journey Download Database with IFUCLEN

The IFUCLEN utility deletes the extracted data from the download database 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 runs as a batch job using DLIBATCH procedure and uses the first database PCB in the PSB as the download database. The ddname used for the download database must match that specified in the DBD and the data set name should reference the one that you wish to use.

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 database 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 database.

```
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=II,STAT=I,C,LTERM=L001
2       OPTIONS APPL=II,KEYRANGE=(10,70),KEY=120,130,LTERM=L002
3       OPTIONS LTERM=L002,DATERANGE=(,19941231)
4       OPTIONS APPL=II,LTERM=L002,KEY=20
5       END
```

Statement	Explanation
1	Deletes all data extracted by terminal L001 for application II; 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 II.
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 II and LTERM L002.
5	The END statement terminates the control statement group.

IFUCLEN JCL Requirements

The IFUCLEN utility runs as a DLIBATCH job and uses the first database PCB in the PSB as the VISION:Journey IMS (DL/I) download database. The ddname used for the download database must match with that specified in DBD; the data set name should reference the one that you wish to use. The download database should be taken offline before running this utility.

II.TCUYCNTL (IMSFTSCL)

The control library member II.TCUYCNTL (IMSFTSCL) contains the following sample JCL for deleting the data from the download database using the IFUCLEN utility.

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, and extraction date (in the form YYYYDDD) of the records deleted.

```
//* VISION:Journey/FTS FEATURE IMS DOWNLOAD DATABASE CLEANUP JCL:
//*   - THIS JOB WILL DELETE THE DATA/REPORT FROM THE DOWNLOAD
//*     DATABASE. THE PARAMETERS IN OPTIONS CONTROL STATEMENT
//*     SPECIFY WHICH DATA/REPORT SHOULD BE DELETED.
//*
//* BEFORE SUBMITTING THIS JOB:
//*   - ADD YOUR INSTALLATION'S JOB STATEMENT.
//*   - VERIFY OR CHANGE THE DSNAMES AS REQUIRED.
//*   - VERIFY OR CHANGE THE DOWNLOAD DATABASE DD NAME AS REQUIRED.
//*   - CHANGE AND VERIFY THE PARAMETERS OF THE OPTIONS CONTROL
//*     STATEMENT TO REFLECT THE DATA WHICH SHOULD BE DELETED.
//*   - THE DBDS AND PSBS MUST ALREADY HAVE BEEN GENERATED.
//*
//*
//*  CLEANUP THE IMS DOWNLOAD DATABASE
//*
//IFUCLEN  EXEC DLIBATCH,MBR=IFUCLEN,PSB=FTSPSBC,RGN=2048K
//STEPLIB  DD
//         DD DISP=SHR,DSN=II.TCUYPGM
//IEFRDER  DD DUMMY,DCB=(BLKSIZE=1408,RECFM=VBS)
//FTSROOT  DD DSN=II.IDXFTS.ROOT,DISP=SHR
//FTSDESC  DD DSN=II.IDXFTS.DESC,DISP=SHR
//FTSRECS  DD DSN=II.IDXFTS.RECS,DISP=SHR
//SYSPRINT DD SYSOUT=*
//DFSVSAMP DD *
2048,6
4096,6
8192,6
//FTSDEL   DD SYSOUT=*
//SYSIN    DD *
 OPTIONS APPL=XXXXXXXX,LTERM=XXXXXXXX,STAT=X,
         KEY=XXXXX,DATE=XXXXXXXX,
         KEYRANGE=(XXXXX,XXXXX),
         DATERANGE=(XXXXXXXX,XXXXXXXX)
 END
/*
```

Figure 5-20 Sample JCL to Maintain the VISION:Journey Download Database

Chapter

6

Programming and Operation Considerations

VISION:Inquiry can execute in four 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
MPP Message Processing Program	II Native VISION:Inquiry
BMP Batch Message Processing	IIBMP Native VISION:Inquiry
Batch (Batch, IMS DL/I batch, and BMP)	IIBATCH Native VISION:Inquiry
TSO	IITSO Native VISION:Inquiry
MPP Message Processing Program	INQEDIT Text Editor
MPP Message Processing Program	AQF
MPP Message Processing Program	VISION:Journey

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 work database size considerations
- Operations considerations for the Intraccess option
- The VISION:Inquiry file transfer operation and the VISION:Journey download database
- 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 IMS and TSO 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, BMP
CUYSHMG	VISION:Inquiry Base Hard-Coded Messages	Batch, TSO BMP, MPP
CUYSHDG	Utility Programs Report Headings	Batch, BMP
CUYXEMSG	Text Editor Hard-Coded Messages	MPP
CUYIEPRM	Text Editor Parameters	MPP
CUYIAMSG	AQF Hard-Coded Messages	MPP
CUYIAMOD	AQF MFS Mod Names	MPP
DIOSQLC	DB2 option independent load module	Batch,TSO BMP, MPP

Program Entry Names

The inquiry programs can be generated with multiple entry points, as described in the following section.

- The means of attachment to DB2 (IMS Attach or CALL Attach)
- For CALL Attach, the DB2 subsystem ID and the plan name
- The type of system database (IMS (DL/I) or DB2)
- For a DB2 system database, the authorization ID and table name
- Under TSO, the application and terminal name assumed if not specified.

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	Batch inquiry program
IXOIDENT	II	MPP inquiry program
IXPIDENT	IIBMP	BMP inquiry program
IXTIDENT	IITSO	TSO inquiry program
IXSIDENT	Utilities IIBATCH II IIBMP IITSO	When not running under IMS If IXBIDENT is not built If IXOIDENT is not built If IXPIDENT is not built If IXTIDENT is not built

The IXSIDENT module can contain only one set of specifications that applies to the utilities and all inquiry processing programs for which an identification module has not been built. However, the inquiry processor identification

modules 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 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 or IMSBATCH 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 (when a DB2 system database is used) 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 6-7).

If the identification module for an 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 MPP, BMP, and IMS Batch inquiry programs ignore the DB2 connection specifications in IXSIDENT and use IMS 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 IMS and TSO Installation Guide*, Chapter 2 "Installation," Step 11 - RECEIVE a USERMOD for the DB2 option and/or the Journey feature to the Global Zone. If you neither have DB2 option nor Journey feature, the default identification module that specifies an IMS system database is used.

To apply the identfication modules to the system, see Applying the Identification module(s) to the system on page 6-8.

The following sections describe the options concerning the identification modules in increasing order of complexity. 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 no other identification modules are used, all VISION:Inquiry programs execute according to its specifications. The only exceptions are:

- The MPP, BMP, and IMS Batch inquiry programs ignore the DB2 connection specifications in IXSIDENT, and use IMS 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.TCUYSRC (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 MPP, BMP, and IMS Batch programs using IMS 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 MPP, BMP, and IMS Batch programs using IMS Attach or from the ordinary batch and TSO programs using CALL Attach with subsystem ID “ssid” and plan name “plan.”

The following operands can be added to IIIDENT macros coded in IXSIDENT or IXTIDENT:

```
[,TSOAPPL=application]
[,TSOTERM=terminal]
```

These operands define the initial application name (simulated transaction code) and terminal name in effect when the TSO inquiry program begins execution.

VISION:Inquiry uses the application and terminal name to determine the directory used to translate inquiries; under TSO, the values specified in these operands are assumed. To change these values, enter the commands:

```
/SET TRAN application
/IAM LTERM terminal
```

If these operands are omitted, the initial application name is II and the initial terminal name is TSO.

Specifying the IIIDENT Macro Options for VISION:Journey

For the Journey feature, the following options are available in the IIIDENT macro:

```
IIIDENT FTS=x,FTSTR=trancode,FTSMB=mfsname
```

Where

FTS=x	Y for VISION:Journey or FTS feature support. N for no VISION:Journey or FTS support. If omitted, the default is N. In this case, all other VISION:Journey parameters are ignored.
FTSTR= trancode	The one to eight character transaction code of the first VISION:Journey program. **Note:** The first VISION:Journey transaction code must end with 1. If omitted, the default is FTS1.
FTSMB= mfsname	The MFSNAME (up to seven characters in length) used in VISION:Journey processing. This screen appears after extraction and before download of data. For VISION:Journey, it automatically starts the download process after this screen displays. If omitted, the default is IDFTSP7. This name must match with the MFS MOD name for the member FTSMFS in the source library.

Building the Identification Modules IXBIDENT, IXOIDENT, IXPIDENT, and IXTIDENT

When any of the identification modules IXBIDENT (batch), IXOIDENT (MPP), IXPIDENT (BMP), or IXTIDENT (TSO) are built and applied to the system, the specifications take precedence over IXSIDENT for the indicated inquiry 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 those inquiry programs for which another identification module has been applied.

These identification modules use 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 (for IXBIDENT), IITSO(for IXTIDENT), II(for IXOIDENT), and IIBMP(for IXPIDENT). These names are the names of the main load modules 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.
- For correct operation of DB2 in MPP regions, the SSID and PLAN operands must be omitted from the IIIDENT macros in IXOIDENT to indicate use of IMS Attach.
- The SSID and PLAN operands may be omitted from IIIDENT macros in IXPIDENT to indicate use of IMS Attach by the BMP program.

If IXOIDENT is assembled, the VISION:Journey specifications take precedence over IXSIDENT for the MPP version of the VISION:Inquiry program. VISION:Journey entries are ignored for all other identification modules.

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 IXOIDENT module with the necessary SMP/E control statements for incorporating into the VISION:Inquiry system.

```
++USERMOD(IQOID01)
  /* USER MOD FOR IXOIDENT IDENTIFICATION MODULE.
  .
++VER(Z038)
  FMID(CCUY650)
  .
++SRC(IXOIDENT) DISTLIB(INQSRCD)
  .
IXOIDENT TITLE 'SYSTEM DATA BASE, VISION:JOURNEY/FTS INFORMATION-MPP'
*
* NOTE:  ALL THE FOLLOWING REFERENCES TO DB2 APPLIES TO THE DB2 OPTION
*        OF THE PRODUCT ONLY.
*
```

Figure 6-1 IXOIDENT Module with Necessary SMP/E control statements (Page 1 of 2)

```
* YOU MAY SPECIFY FROM ONE TO SIXTEEN IIIDENT MACROS.  EACH DESIGNATES
* AN ENTRY POINT INTO THE LOAD MODULE II, 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 AND MUST BE II.
* THE REMAINING ENTRIES WILL BE ALIAS NAMES.
*
* THE STATEMENT " IIIDENT END" IS REQUIRED AS THE LAST SOURCE LINE.
*
*
*  IMS SYSTEM DATA BASE WITH NO VISION:JOURNEY/FTS FEATURE SUPPORT
*
II       IIIDENT TYPE=IMS
*
*  IMS SYSTEM DATA BASE WITH VISION:JOURNEY/FTS FEATURE SUPPORT
*
IIIMSFTS IIIDENT TYPE=IMS,FTS=Y,FTSTR=FTS1,FTSMB=IDFTSP7
*
*  DB2 SYSTEM DATA BASE WITH NO VISION:JOURNEY/FTS FEATURE SUPPORT
*
IIDB2    IIIDENT TYPE=DB2,NAME=DYLINQ.IISYSTEM
*
*  DB2 SYSTEM DATA BASE WITH VISION:JOURNEY/FTS FEATURE SUPPORT
*
IIDB2FTS IIIDENT TYPE=DB2,NAME=DYLINQ.IISYSTEM,                        *
               FTS=Y,FTSTR=FTS1,FTSMB=IDFTSP7
         IIIDENT END
```

Figure 6-1 IXOIDENT Module with Necessary SMP/E control statements (Page 2 of 2)

You can use the following model JCL members provided in the II.PREP.CNTL dataset to RECEIVE, APPLY, RESTORE, and ACCEPT the USERMODs for Idenitfication 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/Datasets.
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 MPP and BMP programs use the transaction code as the application name and the originating LTERM name as the terminal name (unless a user security exit changes this)
- The TSO and batch programs get 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:

- 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
- Whether MFS message formatting is used (and, optionally, a specific output format to be forced)
- Limitations, if any, on sorting
- Whether to use conversational or non-conversational (continuous) mode.

Conversational Mode

In the conversational mode (which is designed for online processing) inquiry processing is controlled by the page length, the maximum number of user database calls for the logical terminal, or a combination of both.

When either a page end or the maximum number of calls 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 may either DEFER the inquiry for later processing or CONTINUE the inquiry for another set of pages of data or another maximum number of calls.

- If the inquiry is deferred, VISION:Inquiry places the checkpoint information into the system database for the logical 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 and begins processing at the point of interruption, until a page end, maximum number of calls, or end of inquiry occurs.

Conversational mode does not work well for batch or BMP inquiries, because there is no "terminal operator" to enter a CONTINUE command if one is needed. However, it is ideal for online use; during the time you take to look over the response to your inquiry and decide what action to take, the IMS message region is available to other users.

The same checkpoint logic (except DEFER command) applies to VISION:Journey during the extraction and writing of the data/report to the VISION:Journey download database by VISION:Inquiry. The DEFER command is not supported for VISION:Journey.

Continuous Mode

Continuous mode is designed for batch or BMP 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 the inquiry cannot be continued. This is not a good choice for use in MPP regions, since resources are used continuously until the entire inquiry is complete.

If continuous mode is used in MPP regions for VISION:Journey, the data is written to the VISION:Journey download database by VISION:Inquiry until the end of data, or until the checkpoint call limit occurs; then, the processing ends and the extraction cannot be continued. You can then download this partial data using VISION:Journey.

The II Program for IMS/DC MPP

The native VISION:Inquiry inquiry processor, II, operates online in an IMS MPP (Message Processing Program) region and executes in the same manner as any other IMS message processing program.

The MPP region JCL may have to be modified to concatenate a DD statement for the PL/I transient library or the LE run-time library and a DD statement for the VISION:Inquiry load library with the STEPLIB libraries.

Access to DB2 is through the IMS Attach. The various requirements for establishing the connection between DB2 and IMS/VS are beyond the scope of this guide.

- If DB2 is already in use in IMS MPP regions at your installation, VISION:Inquiry transactions can be added using the same procedures you would use for any IMS/DC DB2 application.
- Otherwise, you must consult the appropriate IBM documentation for connecting a DB2 subsystem to IMS/DC.

Additionally, the Text Editor facility runs in an online environment in an IMS MPP region and uses an IMS or DB2 work database.

Intraccess runs in an online environment in an IMS MPP region and uses the IMS TCP/IP OTMA (Transmission Control Protocol/Internet Protocol Open Transaction Manager Access) connection (IMS TOC) of IMS Connect to communicate with the PC workstation.

VISION:Journey runs in an online environment in an IMS MPP region and uses an IMS download data base.

The JCL needed in the MPP region can be found in II.TCUYCNTL (IMSJCL).

Terminal Input Format

You invoke the online processor, II, in one of the following ways:

- If the terminal is a video display and MFS is being used, enter the appropriate /FORMAT command. This formats the screen for inquiry.

  To enter inquiries, simply enter the transaction code and the desired inquiry. When using MFS, you have the capability of paging back and forth through the inquiry using the PAGE parameter of MFS (refer to the MFS Utility Manual for additional information). This procedure is repeated for subsequent inquiries.
- If the terminal is a non-MFS terminal, enter the transaction code followed by a blank and the desired inquiry. This process is repeated for each desired inquiry.

Terminal Output Format

Output from VISION:Inquiry appears in one of two formats, each 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. This is 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
```

In vertical format, the order of headings from left to right is the order of the object fields in the inquiry statement. The order of retrieval is determined by the hierarchical structure of the database and is the order of output in horizontal output format.

Horizontal Format

```
EMPLOYEE            = 01401
LAST NAME           = SMITH
SKILL               = CARPENTER
SKILL               = STEAM FITTER
SKILL               = SALESMAN
```

The User Defined Output (UDO) feature of VISION:Inquiry allows you to format the output. You can determine column headings and placement of data and literals on the screen.

The IIBMP Program for IMS/DC BMP

The inquiry processor, IIBMP, operates in an IMS BMP (Batch Message Processing) region.

IIBMP Input

The input of an inquiry to the batch message processor is identical to that of the online processor. When the inquiry is processed by IMS as an input message, it does not cause VISION:Inquiry to be executed but is placed in an IMS message queue. Under the control of the computer operator, VISION:Inquiry is started and processes the waiting messages. VISION:Inquiry processes the inquiry and returns the output to the logical terminal or to an alternate terminal if specified on the inquiry.

IIBMP Output

If the alternate terminal is named "BATCH" the output is written to SYSOUT instead of being sent back to a terminal. The *Advantage VISION:Inquiry for IMS and TSO Release Summary* (previously known as the *Customer Bulletin*) describes an APAR that may be applied to cause all IIBMP output to be routed to SYSOUT.

IIBMP DB2 Access

Access to DB2 may be through CALL Attach or through IMS Attach. The various requirements for establishing the connection between DB2 and IMS/VS are beyond the scope of this manual.

- If DB2 is already in use in batch message processing regions at your installation, VISION:Inquiry transactions can be added using the same procedures you would use for any IMS/DC DB2 application.
- Otherwise, you must consult the appropriate IBM documentation for connecting a DB2 subsystem to IMS/DC or consider using CALL Attach.

IIBMP can run in either continuous or conversational mode. As noted above, continuous mode is usually most appropriate.

Note that input to IIBMP is from a message queue. Inquiries are entered just as they would be for MPP operation. If you wish to process inquiries stored in a data set (or in the job input stream), whether you run in DL/I batch or BMP mode you must use IIBATCH.

Figure 6-2 illustrates example JCL for executing VISION:Inquiry in the BMP (Batch Message Processing) region.

```
//         EXEC IMSBATCH,MBR=IIBMP,PSB=IIPSB04,IN=IIBMP,RGN=512K
//IXXDB    DD DSN=II.IXXDB,DISP=OLD
//IIDBDDM  DD DSN=II.PLANT,DISP=SHR
//IIDBDDMO DD DSN=II.PLANTOV,DISP=SHR
//IIDBDDS  DD DSN=II.SKILL,DISP=SHR
//IIDBDDSO DD DSN=II.SKILLOV,DISP=SHR
//IXXEXTRA DD DSN=&&IXXEXTRA,DISP=(NEW,KEEP),UNIT=3350,
//         DCB=(RECFM=VB,BLKSIZE=2058,LRECL=2054),SPACE=(CYL,(2,2))
```

Figure 6-2 JCL to Execute VISION:Inquiry in the BMP Region

The IIBATCH Program for Batch

The inquiry processor, IIBATCH, operates in ordinary batch, in IMS (DL/I) batch, or in the BMP region.

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 INQIMS DISPLAY PLANT EMP.NO EMP.NAME;;
```

where BATCH is the logical terminal name and INQIMS 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) or in a BMP region (using IMSBATCH). The PSB used is the same as for online or BMP 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.TCUYCNTL (IQBATD and IQBATD2)

The control library member II.TCUYCNTL (IQBATD), as illustrated in Figure 6-3 on page 6-17, can be used as a guideline for VISION:Inquiry batch mode operation to access IMS 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.
//*
//* 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.
//*
//IIBATCH  EXEC DLIBATCH,MBR=IIBATCH,PSB=II,REGION=2500K
//STEPLIB  DD
//         DD   DSN=II.TCUYPGM,DISP=SHR
//         DD   DSN=DSN510.DSNLOAD,DISP=SHR    FOR DB2 TABLES ACCESS
//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 II
 D PLANT PLANT.ID;;
```

Figure 6-3 JCL to Access IMS (DL/I) System and User Databases (IQBATD)

The control library member II.TCUYCNTL(IQBATD2), as illustrated in Figure 6-4, can be used as a guideline for a VISION:Inquiry IMS 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.
//*
//* 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.
//*   - THE SMP/E INSTALLATION MUST BE COMPLETE.
//*
//IIBATCH  EXEC DLIBATCH,MBR=DSNMTV01,PSB=II,REGION=2500K
//STEPLIB  DD
//         DD   DSN=II.TCUYPGM,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
```

Figure 6-4 JCL to Access DB2 System or User Databases in IMS Batch (IQBATD2) (Page 1 of 2)

```
//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 II
D PLANT PLANT.ID;;
```

Figure 6-4 JCL to Access DB2 System or User Databases in IMS Batch (IQBATD2) (Page 2 of 2)

The DDITV02 input file in Figure 6-4 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 system database, user DB2 tables, and user VSAM data sets.

II.TCUYCNTL (IQBATV)

The control library member II.TCUYCNTL (IQBATV), as illustrated in Figure 6-5, 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 DB2 SYSTEM DATABASE & NON-IMS USER DATA SETS.
//*
//* 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.TCUYPGM,DISP=SHR
//         DD   DSN=DSN510.DSNLOAD,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 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 II
 D VSPLANT VSPLANT.ID;;
```

Figure 6-5 JCL to Access Non-IMS (DL/I) System and User Databases (IQBATV)

IITSO Program for TSO

The inquiry processor, IITSO, can access databases in the interactive TSO environment, eliminating the requirement for IMS/DC. This option runs in the TSO time sharing environment, and provides all the features that are available with VISION:Inquiry under IMS/DC. It uses the TSO TGET and TPUT macros and requires that the database access methods be available under TSO.

In order to execute inquiries under TSO, develop a CLIST for authorized users allocating all the necessary IMS (DL/I) files, the system database, and any user databases to be inquired against. Once this is accomplished, VISION:Inquiry can be called and passed the appropriate parameters.

II.TCUYCNTL (DYLIQ)

Control library member II.TCUYCNTL (DYLIQ) contains the following sample CLIST for TSO execution of VISION:Inquiry.

```
       PROC 0 LIST  /* DEBUG USE "CONTROL LIST " */                -
              CLIST /* DEBUG USE "CONTROL LIST SYSLIST CONLIST */   -
              TEST  /* USE TSO "TEST" VERSUS CALL */               -
              BATCH('SYSOUT(A)')     /* "SYSOUT" ALLOCATION */      -
              DUMP('SYSOUT(X) HOLD') /* SYSPRINT PLIDUMP SYSUDUMP */-
              EXTRACT()              /* "IXXEXTRA" DSNAME */        -
              PROGRAM(IITSO)         /* LOAD MODULE TO RUN */       -
              DSNMTV01(DSNMTV01)     /* DB2 MODULE FOR IMS BATCH */ -
              PGMLIB(II.TCUYPGM)     /* LOADLIB */                  -
              DB2LIB(DSN510.SDSNLOAD) /* DB2 LOADLIB */             -
              DDITV02()               /* DB2 INFO FOR IMS BATCH */  -
              DDOTV02()        /* DB2 OUTPUT MSGS FOR IMS BATCH */  -
              IMS(DLI)    /* MAY BE "DLI", "DBB", "BMP", OR "NO" */ -
  /* IMSDB2 PARM BELOW IS ONLY USED IF IMS PARM IS NOT NO"       */ -
              IMSDB2(YES) /* MAY BE "YES" FOR DB2 & IMS, OR "NO" */ -
              RESLIB(IMSVS.RESLIB)   /* IMS LOADLIB */              -
              PSB(II)                /* IMS PSB(NAME) TO USE */     -
              PSBLIB(IMSVS.PSBLIB)   /* IMS PSB LIBRARY */          -
              IMSID()                /* FOR BMP, IMS REGION ID */   -
              DBDLIB(IMSVS.DBDLIB)   /* IMS DBD LIBRARY */          -
              AGN()                  /* REQUIRED BMP PARAMETER */   -
              IMSACB(IMSVS.ACBLIB)   /* IMS ACB LIBRARY */          -
              SSM()                  /* BMP REQUIRED BMP PARAMETER */ -
              IMSACBA(IMSVS.ACBLIBA) /* IMS ALTERNATE ACB LIBRARY */ -
              IMSACBB(IMSVS.ACBLIBB) /* IMS ALTERNATE ACB-LIBRARY */ -
              MODSTAT()             /* DSNAME FOR DDNAME(MODSTAT) */ -
              PROCLIB(IMSVS.PROCLIB)   /* DSNAME FOR DD(PROCLIB) */ -
              DFSVSAMP()               /* DSNAME FOR DD(DFSVSAMP */ -
          DD1(IXXDB)     DB1(II.SYSDB)       /* SYSTEM DATA BASE */ -
          DD2(IIDBDDM)   DB2(II.PLANT.DATA)  /* SAMPLE DATABASES */ -
          DD3(IIDBDDMO)  DB3(II.PLANT.OVFLW)                        -
          DD4(IIDBDDS)   DB4(II.SKILL.DATA)                         -
          DD5(IIDBDDSO)  DB5(II.SKILL.OVFLW)                        -
          DD6()          DB6()                                      -
          DD7()          DB7()  /* DDNAMES AND DSNAMES FOR */       -
          DD8()          DB8()  /* USER DATABASES */                -
          DD9()          DB9()                                      -
          DD10()         DB10()                                     -
          DD11()         DB11()                                     -
          DD12()         DB12()                                     -
```

Figure 6-6 CLIST to Execute VISION:Inquiry in TSO (DYLIQ) (Page 1 of 3)

```
   DD13()         DB13()                                        -
   DD14()         DB14()                                        -
   DD15()         DB15()                                        -
   MAXDD(15) /* MAXIMUM NUMBER OF USER DD STATEMENTS */
IF &STR(&CLIST) = CLIST THEN CONTROL LIST SYMLIST CONLIST
IF &STR(&LIST)  = LIST  THEN CONTROL LIST
IF &STR(&IMSDB2) = YES && ( &STR(&IMS) = DLI | +
  &STR(&IMS) = DBB | &STR(&IMS) = BMP) THEN DO
  SET PROGRAM = &STR(&DSNMTV01)
  IF &STR(&DDITV02) ¬= &STR() +
    THEN ALLOCATE REUSE SHR DD(DDITV02)  DS('&DDITV02')
  IF &STR(&DDOTV02) ¬= &STR() +
    THEN ALLOCATE REUSE SHR DD(DDOTV02)  DS('&DDOTV02')
END
IF        &STR(&IMS) = DLI +
  THEN SET PARM = &STR(DLI,&PROGRAM,&PSB,10,0000)
  ELSE IF &STR(&IMS) = DBB +
  THEN SET PARM = &STR(DBB,&PROGRAM,&PSB,10,0000)
  ELSE IF &STR(&IMS) = BMP +
  THEN SET PARM = &STR(BMP,&PROGRAM,&PSB,,,C00000,,,,,,,,+
                       &IMSID,&AGN,&SSM)
  ELSE SET IMS = NO
SET IILIB = &STR('&PGMLIB')
IF &STR(&DB2LIB)¬=&STR() THEN SET IILIB = &STR('&DB2LIB' &IILIB)
IF &STR(&IMS)   ¬= NO    THEN SET IILIB = &STR('&RESLIB' &IILIB)
CONTROL NOMSG
FREE DD(DFSRESLB MODSTAT)
CONTROL MSG
ALLOC REUSE     DD(SYSOUT)   &BATCH
SET FREE = SYSOUT
ALLOC REUSE     DD(SYSPRINT) &DUMP
ALLOC REUSE     DD(PLIDUMP)  &DUMP
ALLOC REUSE     DD(SYSUDUMP) &DUMP
ALLOC REUSE NEW DD(SORTWK01) CYL SPACE(3 1)
ALLOC REUSE NEW DD(SORTWK02) CYL SPACE(3 1)
ALLOC REUSE NEW DD(SORTWK03) CYL SPACE(3 1)
CONTROL NOFLUSH
ERROR DO
    SET CC = &LASTCC
    ERROR
    CONTROL FLUSH
    WRITE ERROR ENCOUNTERED IN DYLIQ CLIST, LASTCC = &CC..
    IF &STR(&FREE) ¬= &STR() THEN FREE DD(&FREE)
    EXIT CODE(12)
    END
ALLOC REUSE SHR DD(IILIB) DS(&IILIB)
SET FREE = &STR(&FREE IILIB)
IF &STR(&EXTRACT) ¬= &STR() THEN DO
    IF &SUBSTR(1,&EXTRACT) = &STR(.) +
      THEN SET EXTRACT = &STR(&SYSPREF&EXTRACT)
    ALLOCATE REUSE OLD DD(IXXEXTRA) DS('&EXTRACT')
    SET FREE = &STR(&FREE IXXEXTRA)
    END
IF &IMS ¬= BMP THEN DO
    SET I = 1
    DO WHILE &I <= &MAXDD
        SET DD = &&DD&I
        IF &STR(&DD) ¬= &STR() THEN DO
            SET DB = &&DB&I
            ALLOCATE REUSE SHR DD(&DD) DS('&DB')
            SET FREE = &STR(&FREE &DD)
            END
        SET I = &I + 1
        END
    END
IF &STR(&TEST) ¬= TEST THEN SET TEST = CALL
IF &IMS = NO +
```

Figure 6-6 CLIST to Execute VISION:Inquiry in TSO (DYLIQ) (Page 2 of 3)

```
  THEN &TEST '&PGMLIB(IIATTACH)' 'IILIB(&PROGRAM)'
  ELSE DO
    ALLOCATE REUSE     DD(IEFRDER)  DUMMY
    ALLOCATE REUSE SHR DD(PROCLIB)  DS('&PROCLIB')
    ALLOCATE REUSE SHR DD(DFSRESLB) DS('&RESLIB')
    IF &STR(&DFSVSAMP) ¬= &STR() +
      THEN ALLOCATE REUSE SHR DD(DFSVSAMP) DS('&DFSVSAMP')
    IF &IMS ¬= DBB +
      THEN ALLOCATE REUSE SHR DD(IMS) DS('&PSBLIB' '&DBDLIB')
      ELSE IF &STR(&MODSTAT) = &STR() +
          THEN ALLOCATE REUSE SHR DD(IMSACB) DS('&IMSACB')
          ELSE DO
            ALLOCATE REUSE SHR DD(IMSACBA) DS('&IMSACBA')
            ALLOCATE REUSE SHR DD(IMSACBB) DS('&IMSACBB')
            ALLOCATE REUSE SHR DD(MODSTAT) DS('&MODSTAT')
            END
    IF &IMS = BMP && &STR(&DB2LIB) ¬= &STR() +
      THEN ALLOC REUSE SHR DD(DFSESL) DS('&DB2LIB' '&RESLIB')
    &TEST '&PGMLIB(IIATTACH)' 'IILIB(DFSRRC00),&PARM'
    END
IF &TEST ¬= TEST THEN FREE DD(&FREE)
```

Figure 6-6 CLIST to Execute VISION:Inquiry in TSO (DYLIQ) (Page 3 of 3)

Once VISION:Inquiry is invoked, it responds with the following terminal message:

```
ENTER INQUIRY
```

At this point, you specify the transaction code and logical terminal with the following commands.

```
/SET TRAN transaction-name
/IAM LTERM lterm-name
```

When you enter these commands, there is no response. You are free to enter any inquiry against any of the allocated databases, provided they are defined within your directory. The input messages do not require an LTERM or transaction code, just the inquiry.

If the transaction code and LTERM name are not set, VISION:Inquiry uses a default transaction code and terminal name. As supplied, the transaction code is II and the terminal name is TSO; however, these may have been altered during installation. Whatever application and terminal name you use, they must be properly defined in the system database.

When OUTPUT “BATCH” is specified in TSO mode, the SYSOUT data set is opened and the output is written to it instead of to the terminal. The SYSOUT data set is opened only once, so output can be accumulated and printed by a utility when the session is concluded.

VISION:Inquiry prompts you in the same manner it does for online processing.

- As in the batch mode of processing, all inquiry statements must be terminated with two semicolons (;;).
- Single statements, as used in multiple database inquiries, must be terminated with one semicolon (;).
- To terminate the TSO inquiry session enter the following command: /END.

The SORT Command

The SORT command provides you with the ability to reorganize the data you select for display. It can be used in any mode.

When the SORT command is encountered in an MPP inquiry and the inquiry has been syntax checked, it is completed in one of the following ways:

- If MODE=xx0 is specified for the terminal, the inquiry is deferred and a message switch to the transaction specified in the STRAN parameter of the DIRECTORY statement takes place. This transaction is normally a BMP transaction. (If no STRAN is specified, a message switch on the same transaction code is performed. All IMS resources are freed between invocations.)
- If MODE=xx1 is specified for terminal, the inquiry is processed to completion, or until the SLIMIT is reached. PAGE and the maximum number of calls limit do not interrupt processing because the SORT limit parameter determines the number of database calls that are processed.

In all other modes, the inquiry is processed to completion. 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 as follows:

- For batch, the sort files are allocated in the JCL invoking the batch job.
- For batch message processing, the sort files must be allocated in the JCL that invokes the region.
- For TSO, it is recommended that the sort files be made part of a CLIST designed for use with VISION:Inquiry by authorized users. Sort files can be allocated to an MPP region. Each installation should determine whether or not online sorting is permitted because of its impact on system resources. A further discussion on performance is in the next section.

SORT Command Considerations

The impact on performance must be closely studied before allowing sorting in an IMS MPP region. Under normal VISION:Inquiry conversational mode processing, you retrieve only enough data to satisfy a page or maximum number of calls specification. You then lose control of the MPP region, with options to continue, defer, or terminate the inquiry. In this mode, VISION:Inquiry does not allow one inquiry to take control to the exclusion of other users.

However, if sorting is specified in an inquiry being processed online, one inquiry could cause a full database scan to occur. Sorting requires that all data that satisfies the conditions specified be retrieved before it can be displayed. When sorting is allowed in a message region, the inquiry executes in a continuous mode. This results in the IMS MPP region being dedicated to your inquiry until it is complete.

Several options are available to restrict control sorting in the MPP region. These options are as follows:

- The SORT command can be deleted from the vocabulary. This prevents sorting altogether.
- The DD statements for the sort work areas (SORTWKnn) can be omitted in the MPP JCL start-up job. This technique limits the maximum number of records to be sorted because more sorting has to be done in memory and the amount

of memory available to the sort is limited in the MPP. When there is insufficient storage to sort all data for one inquiry, that inquiry is terminated with an error message.

- The inquiry that contains the SORT command can be executed in the BMP by specifying a BMP transaction code, instead of the normal MPP transaction code.
- The MPP inquiry can be re-scheduled to a BMP transaction or a lower priority MPP whenever the SORT command is included in an inquiry. This is done via the STRAN parameter on the DIRECTORY statement and MODE=xx0 on the LTERM statement.
- Coding an SLIMIT on the DIRECTORY statement limits the number of database calls that may be made when a SORT is requested. Exceeding this limit causes the inquiry to be terminated with an error message.

Using 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, except in the MPP region where it is opened and closed for each 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
```

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.

EXTRACT Command Considerations

The EXTRACT command is intended for use in the BMP, BATCH, or TSO environments.

- You can perform extract functions in the MPP region, but you must code DISP=MOD in the control region JCL for the extract data set. DISP=MOD must be specified because VISION:Inquiry opens the data set each time the EXTRACT command is invoked.
- If DISP=MOD is not coded, each EXTRACT command destroys the effect of the previous the EXTRACT command.

However, it is best not to use EXTRACT in an MPP environment because of the data management overhead associated with the opening, closing, and writing of OS files.

Non-MPP users benefit from the fact that VISION:Inquiry opens the extract data set once per execution. Remember, only one BMP copy of VISION:Inquiry is sufficient to service multiple DC users. Since the extract data set is opened once, each EXTRACT command automatically writes extracted information back-to-back in the extract data set. This data set can then be used as soon as the BMP has completed. Note that in MPP mode, the control region data sets are assigned until IMS is brought down.

It is also recommended that the extract data set be a GDG (Generation Data Group), or a data set with disposition of MOD, to allow the BMP to be re-executed following creation of the extract data set.

Your own operational and performance considerations should be evaluated before using the EXTRACT command.

Text Editor Operation

Text Editor runs online in the MPP region and in conjunction with the VISION:Inquiry online version of the native VISION:Inquiry. For executing Text Editor, the DD statements for the Text Editor work database must be added to the MPP region JCL.

Text Editor Screens Considerations

The MFS screens associated with the Text Editor consists of a main screen, help screen, and the transfer screen to the native VISION:Inquiry. The source code for all the menus can be found in the source library (II.TCUYSRC) with the following names.

MEMBER	DESCRIPTION
TEXTMFS	Main Text Editor screen
TEXTHLP	Help screens for TEXTMFS
EDITMFS	Transfer to native VISION:Inquiry screen

The following considerations apply to the Text Editor screens:

- The text of the Text Editor MFS screens can be modified.
- The Text Editor help screen consists of 7 pages. New pages can be added or existing pages can be deleted from the help MFS screen.
- The number of lines and the length of the unprotected fields for the Text Editor main screen, TEXTMFS, can not be changed.
- The PF key functions of the Text Editor main screen, TEXTMFS, cannot be changed or reassigned.
- The source library member II.TCUYSRC (EDITMFS) contains the MFS that formats the screen for the edited query passed by the Text Editor to native mode for processing. This screen is displayed when the command PASS (PF10) or the command SAVE/PASS (PF11) is entered in the Text Editor mode.

  The input area for the EDITMFS screen is 1440 characters, or 20 lines and 72 characters per line. This area is used to transfer the edited inquiry from Text Editor to the native mode. You can change the input area size in the EDITMFS screen, but the size must match with the MAXLEN parameter in the CUYIEPRM member.
- The MOD names for the Text Editor MFS screens must match the parameters in the member CUYIEPRM.

Text Editor Work Database

The Text Editor work database is used to hold intermediate information during the editing process of stored inquiries. The intermediate information for each user is kept separate in the work database and is deleted when a new editing process starts.

The Text Editor work database type is specified as an HDAM IMS (DL/I) database or a DB2 table at installation time.

The Text Editor IMS (DL/I) work database has the same structure as the VISION:Journey download database and can be shared by the two options.

Automatic Query Facility (AQF) Considerations

The Automatic Query Facility (AQF) of VISION:Inquiry, runs online in the MPP region as a conversational program and gives you the capability of creating the inquiries using menus, as opposed to free-form syntax of native VISION:Inquiry.

To execute AQF, the DD statements for the AQF work database must be added to the MPP region.

AQF Screens (Menus) Consideration

AQF consists of a set of main screens (AQFMnn). Associated with each main screen is a help screen (AQFHnn) which describes the function of each menu and the available selection codes.

The following are the source library II.TCUYSRC member names of the main screens and their functions:

MEMBER	DESCRIPTION
AQFM01	Introduction
AQFM02	DB/File Selection
AQFM02A	DB/File View
AQFM03	Field Selection
AQFM03A	Field View
AQFM04	Temporary Field
AQFM05	Qualification
AQFM06	Data Display
AQFM07	Stored Queries

MEMBER <$paranum	DESCRIPTION
AQFM07A	Delete Stored Query
AQFM07B	Stored Queries with Comment
AQFM10	Reentry
AQFIMS	Transfer to Native VISION:Inquiry

The following are the source library II.TCUYSRC member names of the help screens (AQFHnn) and their relation to the main screen:

MEMBER	DESCRIPTION
AQFH01	Introduction
AQFH02	DB/File Select
AQFH03	Field Select
AQFH04	Temp Field
AQFH05	Qualification
AQFH06	Data Display
AQFH07	Stored Query
AQFH10	Reentry

II.TCUYSRC (AQFIMS)

The source library member II.TCUYSRC (AQFIMS) contains the MFS that formats the screen processing for the free-form query generated by AQF. This screen is displayed at the last step of AQF. The screen contains the query generated by AQF and the native VISION:Inquiry transaction code. By pressing the Enter key, the generated query is executed by native VISION:Inquiry and the output is returned.

Making Modifications to the AQD Screens

The following modifications can be made to the AQF screens:

- The MOD names of the AQF main and help screens can be modified. The source library member CUYIAMOD contains the MOD names of the AQF screens. The corresponding MOD names should also be changed in this member, assembled, and linked.
- The number of pages for the help screens can be changed. The corresponding values in the member CUYIAMOD should also be changed, assembled, and linked.
- The text of the help screens can be changed.
- The text of the AQF main screens can be modified.
- The PF key descriptions on the AQF main screens can be repositioned, except for the following:

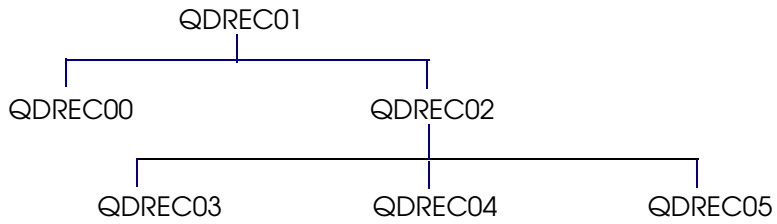
  "PF6=Data Disp" cannot be repositioned and changed in AQFM04, AQFM05, AQFM07, AQFM07B, and AQFM10 screens.

  "PF4=Temp Flds" and "PF5=Qualify" cannot be repositioned and changed in AQFM06, AQFM07, AQFM07B, and AQFM10 screens.
- The PF key functions cannot be changed or reassigned in AQF. For example, the PF1 key is always assigned as a Help key and no other PF key can be used instead.
- If you change the AQF transaction code from IIAQF (default) to any other transaction code, you need to change the PF4 key assignment accordingly in the AQFIMS screen, to be able to return from native VISION:Inquiry to AQF.
- The first logical page of the AQFIMS screen contains 1600 bytes (20 lines) for the input area. This means a generated query with a maximum of 1600 bytes can be passed to native VISION:Inquiry from AQF. The output of the query is displayed starting at line 4. The input area length and the output number of lines can be changed. But the number of output lines should match with the LTERM characteristics defined to VISION:Inquiry (see Chapter 4, "The Definition Process").
- The PF key assignment in the AQFIMS screen can be changed; for example, you can assign PF3 as a return key to AQF instead of PF4.

AQF Work Database Size Considerations

AQF uses a 3-level, 6-segment HIDAM database as a work file. For each user accessing AQF, AQF creates a root segment with the LTERM name as a key, and the associated lower segments. The segments contain information about the screen options selected for a query. Using this information, AQF can rebuild the screens to allow changes to the previous query. If you force AQF to start from the Introduction screen, the old segments will be deleted from the AQF work database and AQF will start from scratch by creating new segments.

The AQF work database is a three level HIDAM database with the following structure:



Note: Because of the different characteristics of the segment types, each segment type of the download database is defined in a separate data set. If you wish, a single data set can be defined for all types instead.

AQF Segment Types

The length of the segments are variable and are minimized based on the number of selected databases, fields, and selection codes. The following is the description of the segment types in the AQF work database:

QDREC01 Control record (root segment), length 256.

QDREC02 One for each database selected (created when the select fields screen is entered).

Minimum length	20 bytes
Maximum length.....	912 bytes
Any sort fields present minimum length	504 bytes
Any list order fields present - minimum length.....	104 bytes
Any control break fields present - minimum length	64 bytes
Any matching KEY fields present - minimum length	24 bytes

The two databases act independently; if all sort fields come from one database, only the QDREC02 for that database will be forced up to 504 bytes.

QDREC03 Contains information for each field of the database. Each record contains information on 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.

Note the FIND and USE databases are treated as if they were two different databases, even if they are really the same database.

QDREC04 Holds temporary field information.

Minimum length..... 460 bytes
Maximum length..... 3460 bytes

Up to 4 temporary fields can be defined per segment.
For each field that temporary field is composed of, add 16 bytes.
For each literal used in a temporary field, add 40 bytes.

QDREC05 Holds qualification data.

Minimum length..... 536 bytes
Maximum length ..... 2536 bytes

Up to 36 lines of qualification held per segment.
Add 56 bytes for each character literal.
Add 24 bytes for each numeric literal.

QDREC00 Holds vocabulary words that AQF needs, and the 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.

Intraccess Considerations

The following section describes 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 an IMS MPP region.

- The VISION:Inquiry queries (inquiries) are generated by Intraccess and passed through IMS TCP/IP OTMA connection facility of IMS Connect to VISION:Inquiry for processing in the IMS MPP region.
- The output of the process will be passed back to Intraccess.

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 data base like other terminals using IIGEN utility.

Programming Considerations

Intraccess passes the query source preceding by the word "Intraccess" to VISION:Inquiry through TCP/IP. VISION:Inquiry checks the first word of the query to find about where the origination of the query is. If the first word of the query is "Intraccess", appropriate flags will be set and then the word "Intraccess" will be removed from the query source.

Using 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 6-10 and Continuous Mode on page 6-11). In continuous mode, the process continues until its completion with no interruption. In conversational mode, the following steps are 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 VISION:Inquiry scratch pad area of the system database.
- All the resources in the IMS related to this operation will be freed.
- Processing resumes for the rest of data automatically until another checkpoint or end of processing occurs.

Downloading Data or a Report with Intraccess

When using the Intraccess option, either data or a report can be downloaded to the PC. The data will be 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/or 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

The file transfer operation involves the following:

- VISION:Inquiry extracts the data or report and writes it to the VISION:Journey download database.
- VISION:Inquiry then starts VISION:Journey. VISION:Journey retrieves the extracted data or report from the download database and downloads it to a PC file.

The following sections discuss the download database and the considerations of the file transfer components. It also contains the VISION:Journey for Windows host considerations. The PC considerations of VISION:Journey for Windows are explained in the *VISION:Journey for Windows System Administrator's Guide.*

VISION:Journey for Windows Operation

VISION:Journey operates online in conversational mode in the MPP region and in conjunction with the online version of native VISION:Inquiry. It needs the following JCL for the download database (see Figure 6-7) to be added to the MPP region JCL.

```
//FTSROOT  DD DISP=SHR,DSN=II.IDXFTS.ROOT
//FTSDESC  DD DISP=SHR,DSN=II.IDXFTS.DESC
//FTSRECS  DD DISP=SHR,DSN=II.IDXFTS.RECS
```

Figure 6-7 JCL for the Download Database

VISION:Journey Online Design

VISION:Journey for Windows is a client/server system, which means you do all interactions on a PC client. When information is required, the PC accesses the server to get requested data. VISION:Journey uses the VISION:Excel, VISION:Inquiry, and VISION:Results host server products as hosts for the client/server relationship.

For VISION:Inquiry, communications between the PC client and enterprise server is achieved via CICS or IMS/DC. The PC portion of VISION:Journey logs on to either CICS or IMS via the 3270 Presentation Space. The 3270 Presentation Space allows VISION:Journey to interact with CICS or IMS exactly the same way as a terminal 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 invokes VISION:Inquiry through transactions for processing the inquiry, extracting the data, and writing it with the necessary control information to an intermediate download file. The download file is used to transfer data and reports to the PC client.

VISION:Journey Sequence of Actions

Using IMS transactions, VISION:Journey, along with VISION:Inquiry, performs the following sequence of actions:

1. VISION:Journey logs on to IMS.
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 command PCE or OUTPUT 'dummy terminal' which initiates the download process.
4. VISION:Inquiry starts the first VISION:Journey transaction (FTS1) which calls the DYLI0SS program. The FTS1 transaction sends the MFS IDFTSP7 along with the control information to start the download.
5. VISION:Journey invokes the FTS3 transaction to download the data or report to the PC. The FTS3 transaction calls the "send/receive" program (DYLI010) which, in turn, calls the "read/write" program (DYLI020) to read the output that is to be downloaded from the download file. The "send/receive" program then calls the compressed program (DYLI030) to compress the data. The compressed data is sent onto the screen, which is the 3270 Presentation Space. The PC side of VISION:Journey captures the data and decompresses it. The FTS3 transaction ends once all the data has been downloaded.
6. VISION:Journey deletes the data or report in the IMS (DL/I) download file.
7. VISION:Journey logs off the host.

VISION:Journey Host Considerations

The following considerations apply to the installation and maintenance of VISION:Journey for Windows:

- The default MFS screen ODYMFS 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 screen. 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.
- Any VISION:Inquiry transaction (application) which uses VISION:Journey 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"). It is defined in the system database like other terminals using IIGEN utility.
- VISION:Journey uses a MFS screen (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 delete the data in the download file after completion of download.

VISION:Journey Download Database

The VISION:Journey download database is a staging file used to hold the data or report for download processing. The data or report is assigned a terminal/user ID along with a unique subsequence number to differentiate it from other data.

The download database is an HDAM IMS (DL/I) database and should be present during file transfer. The download database records consist of a 10-byte key. The following are the fields with their size and type that comprise the key:

Terminal name (8 characters)	VISION:Journey terminal identification. This is the same name used to define terminals to VISION:Inquiry.
Subsequence (2-byte binary)	A unique number assigned to each set of extracted data. The number is used to distinguish one set of extracted data from another.

The structure of the VISION:Journey download database is as follows:

Dummy control record	A dummy record which is used to initialize the download database. There is only one of this type of record in the data set.
Master control record	A record created only once for each user/terminal, the first time the user/terminal executes the VISION:Journey command. It contains the next subsequence number to be assigned to download data. It also contains "code points" required for transmission of data between the server and PC. The subsequence number of this record is zero.
Root record	Contains information about the extracted data. There is one root record for each extraction.
Data record	Data records are the actual data to be downloaded to the PC.
Description record	Contains information about each field of a data 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 extraction of a formatted report, as opposed to data, one description record is created.

The VISION:Journey download database can also be defined to VISION:Inquiry as a user database. Its contents can be displayed using the VISION:Inquiry DISPLAY command by the System Administrator to obtain information about the

extracted data or report. Because different VISION:Journey download database record types exist, you need to use an IF clause with the DISPLAY command to correctly display the information about each record type.

II.TCUYSRC (IIDMGEN)

The source library member II.TCUYSRC (IIDMGEN) contains the MAPGEN for the download database.

Figure 6-8 shows the MAPGEN used to define the IMS (DL/I) download database to VISION:Inquiry as a user database. The MAPGEN does not include nor require the internally used Master Control Record.

```
*
*        MAPGEN FOR VISION:Journey/FTS IMS DOWNLOAD DATABASE
*
 MAPGEN  MAP=FTSMAP,DBD=IDXFTS,NAME=FTSDB,
         DESC="VISION:Journey/FTS FEATURE DOWNLOAD DATABASE"
 SEGMENT SEGM=FTSROOT,PARENT=0,BYTES=2400,KEY=ROOTKEY,TYPE=V
 FIELD   START=3,LENGTH=10,TYPE=C,NAME=ROOTKEY,KEY=EQUAL,
         DESC="ROOT KEY, CONSISTS OF TERMINAL NAME AND SUBSEQUENCE #"
 FIELD   START=3,LENGTH=8,TYPE=C,NAME=TERMNAME,
         DESC="TERMINAL NAME INITIATED THE EXTRACTION"
 FIELD   START=11,LENGTH=2,TYPE=B,NAME=SUBSEQ,
         DESC="UNIQUE SUBSEQUENCE NUMBER"
 FIELD   START=13,LENGTH=8,TYPE=C,NAME=APPLNAME,
         DESC="APPLICATION NAME INITIATED THE EXTRACTION"
 FIELD   START=37,LENGTH=4,TYPE=B,NAME=RECCOUNT,
         DESC="THE NUMBER OF RECORDS EXTRACTED"
 FIELD   START=41,LENGTH=4,TYPE=P,NAME=CREDATE,
         DESC="CREATION DATE IN YYYYDDD FORMAT"
 FIELD   START=45,LENGTH=4,TYPE=P,NAME=CRETIME,
         DESC="CREATION TIME IN HHMMSST FORMAT"
 FIELD    START=49,LENGTH=1,TYPE=C,NAME=CRESTAT,
         DESC="STATUS OF EXTRACTED DATA"
 FIELD   START=50,LENGTH=3,TYPE=C,NAME=CREENV,
         DESC="FILE TRANSFER ENVIRONMENT, IMS OR CIS FOR CICS"
 FIELD   START=53,LENGTH=2,TYPE=B,NAME=CON#LEN,
         DESC="PC FILE NAME LENGTH"
 FIELD   START=55,LENGTH=255,TYPE=C,NAME=CONSTANT,
         DESC="PC FILE NAME"
 SEGMENT SEGM=FTSDESC,PARENT=FTSROOT,BYTES=82,KEY=FIELDSEQ,TYPE=V
 FIELD   START=5,LENGTH=2,TYPE=B,NAME=FIELDSEQ,KEY=SEQ-U,
         DESC="FIELD SEQUENCE NUMBER STARTING FROM 1"
 FIELD   START=7,LENGTH=2,TYPE=B,NAME=FIELDLOC,
         DESC="RELATIVE LOCATION OF THE FIELD"
 FIELD   START=9,LENGTH=2,TYPE=B,NAME=FIELDLEN,
         DESC="FIELD LENGTH - 1"
 FIELD   START=11,LENGTH=2,TYPE=B,NAME=EXLEN,
         DESC="OUTPUT LENGTH OF THE FIELD - 1"
 FIELD   START=13,LENGTH=1,TYPE=C,NAME=FIELDTYP,
         DESC="FIELD TYPE"
 FIELD   START=15,LENGTH=2,TYPE=B,NAME=FLDSCALE,
         DESC="FIELD SCALING FACTOR"
 FIELD   START=17,LENGTH=2,TYPE=B,NAME=NAME#LEN,
         DESC="LENGTH OF FIELD NAME"
 FIELD   START=19,LENGTH=32,TYPE=C,NAME=FLD#NAME,
```

Figure 6-8 IMS MAPGEN for Download Database (Page 1 of 2)

```
        DESC="FIELD NAME"
SEGMENT SEGM=FTSRECS,PARENT=FTSROOT,BYTES=8010,KEY=RECORDSQ,TYPE=V
FIELD   START=5,LENGTH=4,TYPE=B,NAME=RECORDSQ,KEY=SEQ-U,
        DESC="RECORD SEQUENCE NUMBER STARTING FROM 1"
FIELD   START=9,LENGTH=2,TYPE=B,NAME=RECLN,
        DESC="RECORD LENGTH"
FIELD   START=11,LENGTH=50,TYPE=C,NAME=RECORD,
        DESC="THE FIRST 50 BYTES OF EXTRACTED RECORD"
FINISH
```

Figure 6-8 IMS MAPGEN for Download Database (Page 2 of 2)

Description of the Fields

The following is the description of each field, grouped by segment.

FTSROOT Segment

The FTSROOT segment contains the general information about each extraction.

TERMNAME	Terminal name which initiated the extraction
SUBSEQ	Unique subsequence number assigned to the extraction
APPLNAME	Application name
RECCOUNT	Number of data records extracted (number of occurrences of FTSRECS segment)
CREDATE	Creation date (YYYYDDD)
CRETIME	Creation time (HHMMSST)
CRESTAT	Extracted data status: F=completed C=checkpoint taken I=incomplete due to some errors
CREENV	Environment (MPP for IMS version, CIS for CICS version)
CON#LEN	PC file name length
CONSTANT	PC file name; the name specified as a literal following the VISION:Inquiry command used for file transfer

FTSDESC Segment

The FTSDESC segment contains information about each field to be extracted. The number of occurrences of this segment is equal to the number of fields to be extracted for each extraction.

FIELDSEQ	Internal field sequence number starting from 1
FIELDLOC	Relative location of field within the record
FIELDLEN	Field length - 1
EXLEN	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"for more information on MAPGEN).
FIELDTYP	Type of field. The same as the type parameter used in the field statement of MAPGEN (see Chapter 4, "The Definition Process").
FLDSCALE	Scaling factor. The same as the scale parameter used in the field statement of MAPGEN (see Chapter 4, "The Definition Process").
NAME#LEN	The length of the field name
FLD#NAME	Field name

FTSRECS Segment

The FTSRECS segment contains the actual data extracted from the user database/file. The number of occurrences of this segment is equal to the number of data records for each extraction.

RECORDSQ	Internal record sequence number starting from 1
RECLN	Record length
RECORD	Data record. The data record can be up to 8000 bytes. In Figure 6-8, the record shows only the first 50 bytes of the record.

The structure of the VISION:Journey download database is the same as the Text Editor IMS (DL/I) work database and can be shared by the two options.

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 IITSO Program for TSO on page 6-21 are also applicable. This facility allows SQL SELECT statements combined with some VISION:Inquiry commands to be processed against DB2 tables and 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 about DB2 tables and views accessed by the native SQL syntax facility is extracted from the DB2 catalog directly. No MAPGEN is required to define the tables and views in the system database.
- The output is 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 cases where there are no column names, such as expressions, a period is displayed as the column heading.
- The length of the edited column values in the output report 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
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 are 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, the null value of a character column with a length of 4 is displayed as ????. However, through the output user exit (see Chapter 7, "User Exits"), you can specify what is displayed for the null value.
- The null value is shown as X'00' in the extract data set.
- The floating point and graphic values can be processed only by an output user exit (see Chapter 7, "User Exits"). If no output exit is used, VISION:Inquiry issues 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.
- Native SQL syntax can be used with the EXTRACT command with the same considerations discussed in Using the EXTRACT Command on page 6-27 through EXTRACT Command Considerations on page 6-30.
- Native SQL syntax can be used with the Text Editor with the same considerations discussed in Text Editor Operation on page 6-31 through Text Editor Work Database on page 6-32.
- The inquiries are 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 with the same considerations discussed in The SORT Command on page 6-25.
- Native SQL syntax is not supported in AQF mode.
- Native SQL syntax is not supported with VISION:Journey for Windows.

Chapter

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.

IXXUIN, IXXUSEC, and IXXUOUT 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 7-6, and PL/I Exit Considerations on page 7-9. 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 7-3 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 dataset 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/Datasets.
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(CCUY650).
++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):

- The first parameter is the list of PCB addresses from the PSB passed to VISION:Inquiry by IMS. In batch, if the inquiry program is not executing under IMS, 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 O = MPP P = BMP T = TSO
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 or DB2
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 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 IMS and TSO 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 fullword containing the message number, a pointer to the substitution text, and a halfword 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 7-8 shows how you may define a 20-byte area in DSA to be used in your user exit.

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
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
```

Figure 7-2 IXUSER Macro Source Code (Page 1 of 2)

```
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
*
*               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 2 of 2)

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.

Example Assembler Input Exit

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.

```
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
*
*
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

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 7-4 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 7-4 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	LTERM

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.

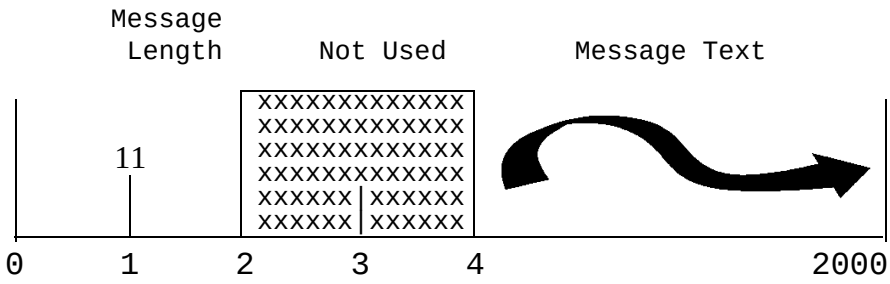
- The first two bytes of the message input area contain the length of the message.
- Bytes 5 through 2000 contain the message text.

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.

LTERM The address of the 8-byte logical 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 7-16.
- Otherwise use IXXUOUT as described in section IXXUOUT for Inquiries with no Native SQL Statement on page 7-12.

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 7-16 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 7-5 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 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 7-6.)

- 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 7-6 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/LTERM 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	PSB
4	EXIT INDICATOR
8	TRANSACTION
12	LTERM
16	USER RETURN ARGUMENT 1
20	USER RETURN ARGUMENT 2

IXXUSEC 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'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/LTERM 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 LTERM name against the LTERMs 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 LTERM against all LTERMs 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.

LTERM

The address of the 8-byte logical terminal name.

USER RETURN ARGUMENT 1

The address of an 8-byte area where the exit routine is to place the APPL name (transaction code) 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.

Appendix

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'
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’

Appendix

B

Sample User Exits

Sample User Exits

The installation source library, II.TCUYSRC, contains exit samples in Assembler and in PL/I. Exit names and exit sample names are:

Exit Name	Exit Sample Name
IXXUCON	IXXUCONP IXXUCONS
IXXUFNC	IXXUFNCS
IXXUIN	IXXUINS
IXXUOUT	IXXUOUTS
IXXUSEC	IXXUSECS

The source library member names for these samples begin with the characters 'IXXU'. With the exception of IXXUCONP (P for PL/I), the exit samples end with the letter S.

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.TCUYSRC (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               */
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 2 of 4)

```
/*       CHARACTER TO HEX CONVERSION                          */
/*===========================================================*/
   IF EXIT_INDICATOR.BYTE (2) = '01000000'B THEN
      DO;
         CALL INT_EXT_CONV;
         RETURN;
      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;
```

Figure B-1 IXXUCONP Source (PL/I) - IXXUCON Conversion Exit (Page 3 of 4)

```
   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;
      /*=====================================================*/
      /* 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.TCUYSRC (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
```

Figure B-2 IXXUCONS Source (Assembler) - IXXUCON Conversion Exit (Page 1 of 3)

```
         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
*        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'
```

Figure B-2 IXXUCONS Source (Assembler) - IXXUCON Conversion Exit (Page 2 of 3)

```
         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
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.

LABEL	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
*
```

Figure B-3 IXHEX Macro Source (Page 1 of 3)

```
             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
*
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
```

Figure B-3 IXHEX Macro Source (Page 2 of 3)

```
             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 3 of 3)

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,
```

Figure B-4 FIELD Statements Using the Sample Conversion Exit (Page 1 of 2)

```
     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 (Page 2 of 2)

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

Figure B-6 Output Generated from the Conversion Exit in INQUIRY1 (Page 1 of 2)

```
                                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
                                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 D7CBEB03D3C9E240D3D6C3D2D4C5EBC5D94040404040404040  48,000.00  004800000C
                                                                                    59,000.00  005900000C
         10105  MARY ANN THOMAS   04C1D9EB40CID5D540E3C8D6D4CIE24040404B404B40404040  15,600.00  001560000C
20150    21116  WILMA FORD        E6C9D3D4C140C6D6D9C4404040404040404040404040404040  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      D7C5E3C5D940E9C1E3D2C9D5404040404040404040404040404040  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
```

Figure B-7 Output Generated from the Conversion Exit in INQUIRY2 (Page 1 of 2)

```
                                                                                                  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         ESC9C3D2EB40E6C1D9C4404040404040404040404040404040          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 (Page 2 of 2)

```
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
IXXF      TITLE '*** USER FUNCTION ROUTINE: GENERIC - IXXUFNC ***'
IXXUFNC   IXBEGIN TYPE=USER
          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
```

Figure B-9 IXXUFNC Source - IXXUFNC Function Exit (Page 1 of 3)

```
*
* CHECK FUNCTION NAME
*
          SPACE
          L      R1,AFD                     POINT TO FUNCTION DESCRIPTOR
          LA     R5,FDTABLE                 ADDRESS OF ALLOWABLE FUNCTIONS
          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
          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
*
         LOAD    EPLOC=FDTADDR             LOAD MODULE
*
         LR      R15,R0                    ADDRESS OF ENTRY POINT
*
         CALL    (15)                      BRANCH TO MODULE
*
         DELETE  EPLOC=FDTADDR             DELETE MODULE
*
         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
```

Figure B-9 IXXUFNC Source - IXXUFNC Function Exit (Page 2 of 3)

```
         IXRETURN RC=(15)                  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
TABLEFD  DSECT
FDTNAME  DS      CL32
FDTADDR  DS      A
FDTMOD   DS      A
FDTLTH   DS      A
         SPACE   3
         IXUSER
         SPACE   3
          END
```

Figure B-9 IXXUFNC Source - IXXUFNC Function 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.

or

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
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
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 1 of 4)

```
             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
*
             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
*
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 2 of 4)

```
                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
                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
```

Figure B-11 IXXUFTM Bit Testing Function Source (Page 3 of 4)

```
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 XXUFTM user function is:

%TMname=USER(Fieldname n 'xxxxxxxx')

The XXUFTM 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. Assembled 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 Chapter 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.

```
         PRINT NOGEN
UIN      TITLE '*** USER INPUT EXIT: IXXUIN - SECURITY ROUTINE ***'
IXXUIN   IXBEGIN TYPE=USER
         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
```

Figure B-15 IXXUINS Source - IXXUIN Input Exit (Page 1 of 5)

```
         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
         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       R8,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
```

Figure B-15 IXXUINS Source - IXXUIN Input Exit (Page 2 of 5)

```
         LTR     R7,R7                IS CSECT LINK EDITED?
         BNZ     TABLEIN              YES, BYPASS
*
         LOAD    EP=IXXSTABL
*
         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
         L       R8,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
*
```

Figure B-15 IXXUINS Source - IXXUIN Input Exit (Page 3 of 5)

```
         DELETE EP=IXXSTABL            DELETE MODULE
*
         XC      ATABLE,ATABLE         RESTORE VCON TO ZERO
         SPACE   5
RETURN   DS      0H
         EJECT
         SPACE   25
*********************************************************************
*                                                                   *
*             ADDITIONAL CODE CAN BE APPENDED HERE                  *
*               PLEASE BEWARE OF DUPLICATE LABELS                   *
*                                                                   *
*********************************************************************
         EJECT
*
* RETURN TO CALLER
*
         SPACE
         IXRETURN
         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(R8)
MVCBLNK  MVC     0(0,R8),=CL8' '
         EJECT
*
* AREAS AND CONSTANTS
*
         SPACE
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
```

Figure B-15 IXXUINS Source - IXXUIN Input Exit (Page 4 of 5)

```
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      CL2000                  MESSAGE
         SPACE   5
         IXUSER
         SPACE   5
         END
```

Figure B-15 IXXUINS Source - IXXUIN Input Exit (Page 5 of 5)

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     TRANCODE: IISY     INQUIRY:
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 code (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.

Appendix

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
```

Figure C-2 IXSGEN Macro Source Code (Page 1 of 3)

```
                DC      AL2(&PASSN,PASSWELN)   NUMBER OF PASSWORDS
.*                                             LENGTH OF TABLE ELEMENT
                SPACE   5
                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
```

Figure C-2 IXSGEN Macro Source Code (Page 2 of 3)

```
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                      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)

Appendix

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.

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
II	IMS Message Processing	PL/I and Assembler
IIBMP	IMS Batch Message Processing	PL/I and Assembler
IIBATCH	Batch, DL/I Batch, or BMP	PL/I and Assembler
IITSO	TSO or DL/I under TSO	PL/I and Assembler
IIATTACH	TSO	Assembler
IIINIT	Batch or DL/I Batch	PL/I
IIGEN	Batch, DL/I Batch, or BMP	PL/I
IXUIQRY	Batch or DL/I Batch	PL/I
IXUSQRY	Batch or DL/I Batch	PL/I
IXULOAD	Batch or DL/I Batch	PL/I

Load Module Name	Execution Environment	Programming Language
IXUSTAT	Batch or DL/I Batch	PL/I
IXUUNLD	Batch or DL/I Batch	PL/I
IIDEMO	DL/I Batch	PL/I
IVDEMO	Batch	PL/I
IXXRMODL	DL/I Randomizer	Assembler
CUYSHDG	Batch, DL/I Batch, or BMP	Assembler
CUYSHMG	Batch, DL/I Batch, IMS Message processing, or BMP	Assembler

Text Editor Modules

The Text Editor consists of the following executable modules which reside in the installation load module library.

Load Module Name	Execution Environment	Programming Language
INQEDIT	IMS Message Processing	PL/I and Assembler
IFUCLEN	Batch or DL/I Batch	PL/I
IFUINIT	DL/I Batch	PL/I
CUYXEMSG	IMS Message processing	Assembler

AQF Modules

The AQF portion of the system consists of the following executable programs which reside in the installation load module library.

Load Module Name	Execution Environment	Programming Language
IAOINIT	DL/I Batch	PL/I
IAOI01	IMS Message Processing	PL/I
CUYIAMSG	IMS Message processing	Assembler
CUYIAMOD	IMS Message processing	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, DL/I BATCH, IMS Message Processing, or BMP	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
DYLI0SS	IMS Message Processing	Assembler
DYLI010	IMS Message Processing	Assembler
DYLI020	IMS Message Processing	Assembler
DYLI030	IMS Message Processing	SAS/C
IFUCLEN	DL/I Batch	PL/I
IFUINIT	DL/I Batch	PL/I

Appendix

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.TCUYCNTL	II.DCUYCNTL	Sample JCls for Post-Installation and maintenance of the system
II.TCUYMAC	II.DCUYMAC	Macros for customization, exit routines, and building MFS screens
II.TCUYPGM	II.DCUYPGM	VISION:Inquiry load modules
II.TCUYSRC	II.DCUYSRC	Source for various control blocks, data for test systems, source for MFS screens, and error messages, and vocabulary
	II.DCUYOBJ	VISION:Inquiry object modules in the link-edited format
II.TCUYCLST	II.DCUYCLST	CLISTs for Post-Installation Dialog
II.TCUYMLIB	II.DCUYMLIB	Messages for Post-Installation Dialog
II.TCUYPLIB	II.DCUYPLIB	Panels for Post-Installation Dialog
II.TCUYSLIB	II.DCUYSLIB	Skeleton JCLs for Post-Installation Dialog

Source Library

The source library contains the following members

AQFHnn	Source for generating MFS formats for AQF
AQFIMAn	MFS copy library members used in generating MFS formats for AQF screens (AQFIMS)
AQFIMS	Source for generating MFS formats for AQF
AQFIMS2	MFS copy library member used in generating MFS formats for AQF screens (AQFIMS)
AQFMnn	Source for generating MFS formats for AQF
AQFMnnA AQFMnnB AQFPSB	Sample PSB for using AQF
AQFPSBIN	Sample PSB for initializing the AQF work database using the IAOINIT utility
DB2CATA	DB2 catalog program
DJCSECT	VISION:Journey load module names (DYLI010, DYLI020, and DYLI030) CSECT
EDITMFS	Source for generating MFS formats for the Text Editor
FTSMFS	Source for generating MFS formats for VISION:Journey
FTSPSBC	Sample PSB for cleanup of the Text Editor IMS (DL/I) work database and the VISION:Journey download database
FTSPSBL	Sample PSB for initializing the Text Editor IMS (DL/I) work database and the VISION:Journey download database
CUYIAMSG	AQF messages
CUYIAMOD	MFS MOD names for AQF
IDXFTS	Sample DBD for the Text Editor IMS (DL/I) work database and the VISION:Journey download database
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
IIIMSnn	MFS copy library members used in generating MFS formats for INQIMS, the Text Editor screen (EDITMFS), and the AQF screen (AQFIMS)
IIMSGnn, IIPRTnn	MFS copy library members used in generating MFS formats for INQIMS, INQUDO, the Text Editor screen (EDITMFS), and the AQF screen (AQFIMS)
IIPSB	Sample PSB for inquiries against the test 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
IIUDOnn	MFS copy library members used in generating MFS formats for INQUDO
IIVOCAB	Standard system vocabulary
INQIMS	Source for generating MFS formats for native VISION:Inquiry
INQUDO	Source for generating MFS formats for native VISION:Inquiry with the UDO feature
IXBIDENT	Source for DB2 and system database identification module
CUYGCHK	IIGEN checkpoint interval control module
CUYIEPRM	Text Editor parameters
IXOIDENT	Source for DB2, VISION:Journey, and system database identification module
IXPIDENT	Source for DB2 and system database identification module
CUYSHDG	Utility program headings
CUYSHMG	Hard-coded messages
IXSIDENT	Source for DB2, VISION:Journey, and system database identification module
IXTIDENT	Source for DB2 and system database identification module

IXXAQF IXXAQFIX	Sample DBDs for AQF HIDAM work database and its index
IXXDB	Sample DBD for IMS (DL/I) system database
CUYXEMSG	Text Editor error messages
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
JRNPSB1 JRNPSB2	Sample PSBs for VISION:Journey
ODYMFS	Source for generating MFS formats for VISION:Journey
TEXTHLP TEXTMFS	Source for generating MFS formats for the Text Editor
TEXTPSB	Sample PSB for using Text Editor
TXTIMSnn	MFS copy library members used in generating MFS formats for the Text Editor screen (EDITMFS)
VSDATAH VSDATAK VSDATAR	Data used to load the VSAM test data sets

Control Library

The control library contains the following members

AQFINIT	JCL to allocate and initialize the AQF work database
ASMCHKI	JCL to assemble the IIGEN checkpoint interval control module, CUYGCHK
ASMEMSG	JCL to assemble the Text Editor messages, CUYXEMSG
ASMEPARM	JCL to assemble the Text Editor parameters, CUYIEPRM
ASMHMSG	JCL to assemble the hard-coded messages module, CUYSHMG
ASMOMODS	JCL to assemble the MFS MOD names, CUYIAMOD, for AQF
ASMOMSG	JCL to assemble the AQF messages, CUYIAMSG
ASMSHDG	JCL to assemble the utility headings module, CUYSHDG
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
DB2TXTCL	JCL to execute the IFUCLEN cleanup utility for the Text Editor DB2 work database

DB2TXTIN	SPUFI input to create and initialize the Text Editor DB2 work database
DB2UNLD	JCL to unload a DB2 system database using the IXUUNLD utility
DYLIQ	The model CLIST for running VISION:Inquiry in a TSO environment
IIFORMAT	JCL to build MFS formats
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
IMSFTSCL	JCL to execute the IFUCLEN cleanup utility for the VISION:Journey download database
IMSFTSIN	JCL to allocate and initialize the VISION:Journey download database
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
IMSJCL	JCL to be added to your MPP region JCL
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
IMSTXTCL	JCL to execute the IFUCLEN cleanup utility for the Text Editor IMS (DL/I) work database
IMSTXTIN	JCL to allocate and initialize the Text Editor IMS (DL/I) work database
IMSUNLD	JCL to unload an IMS (DL/I) system database using the IXUUNLD utility

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
VSAMCOBL	JCL to run the COBOL converter program for VSAM data sets
VSAMDEMO	JCL to allocate and load the VSAM non-hierarchical test data sets, VSPLANT and VSSKILL
VSMHDEMO	JCL to allocate and load the VSAM hierarchical test data sets, VSHPLANT and VSHSKILL

Index

Symbols

$$$$$IXX, 3-5, 5-17
/END, 6-24
/FORMAT, 6-12
/IAM LTERM, 4-50, 6-10
/NOSEQ, 6-16
/SEQ, 6-16
/SET BATCH, 6-15
/SET ONLINE, 6-15
/SET TRAN, 6-10, 6-15

A

access, 1-3, 2-2, 6-10
  databases, 6-10
  fields, 6-10
Acrobat Reader, 1-1
  using, 1-2
Adobe Acrobat Reader, 1-1
alternate indices, 2-2, 4-9, 4-33
  KEY parameter, 4-23
  PATH parameter, 4-33
alternate keys, 4-43
APPL statement, 4-5, 4-7
  NAME parameter, 4-7
  NAMELENGTH parameter, 4-7, B-8
  syntax, 4-7
APPL/END control statement group, 4-68
applications, 2-2, 3-5, 3-9, 4-1, 4-53
  $$$$IXX, 3-5, 3-6
  define with IIGEN, 4-7
  maintain, 4-66
  names, 4-6
AQF (Automatic Query Facility), 1-3, 1-6, 2-2
  display description, 4-14, 4-24
  executable load modules, D-3
  IIAQF transaction code, 6-34
  MFS screens, 2-4
  online environment, 6-2
  processing, 2-11
  screen modifications, 6-34
  screens (menus), 6-32
AQF work database, 2-4, 2-14, 2-16, 3-10, 5-1
  considerations, 6-34
  initialize with IAOINIT, 2-16, 5-1, 5-6
  segments, 6-35
  size, 6-34
  structure, 6-35
  utilities, 5-1
AQFHnn, 6-33
AQFIMS, 6-33, 6-34
AQFINIT, 5-48
AQFMnn, 6-32
AQFPSBIN, 5-48
Assembler, 7-2, 7-6, B-1
  user exits, 7-2
attach, 6-4

CALL Attach, 5-8, 6-4, 6-14, 7-4
IMS Attach, 6-4, 6-12, 6-14
AVERAGE, A-1

B

batch, 1-6, 6-11
execution, 2-5, 2-9
JCL, 5-5, 6-17
BMP (Batch Message Processing), 1-6, 6-11
execution, 2-5, 2-8
IIBATCH, 6-15
IIBMP, 6-14

C

CA-Librarian, 4-28
CA-Panvalet, 4-28
CD-ROM contents, 1-1
checkpoints, 3-10, 5-22
ignore conditions, 4-51
page end, 4-50
page limit, 4-51, 4-53
scratch pad, 3-10, 6-10
time limit, 4-50
CLIST, 6-21
COBOL, 3-15
copy books, 4-26, 4-28, 4-31
hierarchical record layout, 3-15
COBOL converter, 4-26, 4-34
CA-Librarian, 4-28, 4-29, 4-32, 4-34
CA-Panvalet, 4-28, 4-29, 4-32, 4-34
create field definitions, 4-26
features not supported, 4-35
for IMS (DL/I) databases, 4-26, 4-27, 4-30
for VSAM data sets, 4-26
for VSAM non-hierarchical data sets, 4-32
JCL (IMSCOBL), 4-27
JCL (VSAMCOBL), 4-29
user responsibility, 4-36
COBPLANT, 4-34
code points, 6-41
Computer Associates, 1-2
connected directories, 3-8, 3-9, 4-46
transaction code, 4-2
considerations, 3-2
AQF menus, 6-2
AQF work database, 6-2
COBOL converter (IMS), 4-35
COBOL converter (VSAM), 4-36
continuous mode, 6-11
conversational mode, 6-11
EXTRACT, 6-2, 6-30
IIBMP, 6-14
IXXUCON, 7-22
performance, 3-2
SORT, 6-2
sort, 6-26
storage, 7-2
user exits, 7-2, 7-9
contacting, 1-2
Computer Associates, 1-3
Computer Associates web page, 1-3
Total License Care (TLC), 1-2
CONTINUE, 6-10, 6-11, A-1
continuous mode, 1-3, 6-11, 6-38
control statement groups, 4-5
APPL/END, 4-68
DELETE, 4-5, 4-71
DIRECTORY, 4-5, 4-44
ERRLOAD, 4-5, 5-4, 5-12, 5-14
LTERM, 4-5, 4-49
MAPGEN, 4-5, 4-9
OPTIONS, 5-5, 5-6, 5-24
SYSTEM, 4-5, 5-4, 5-17
UPDATEDIR, 4-5, 4-68
control statements
comments, 5-3
page ejects, 5-3
conversational mode, 1-3, 6-10, 6-38

conversion, 4-21, B-2
format of field, 4-24
IXHEX macro, B-8
IXXUCONP, B-2
IXXUCONS, B-5
output field, 4-22
COUNT, A-1

D

database map names, 4-2
database maps, 2-14
databases, 1-6
DB2, 1-3, 1-6, 2-1, 2-2, 3-6, 4-13
define, 4-9
HDAM, 1-7, 3-1
IMS (DL/I), 1-3, 1-6, 2-1, 2-2, 3-6, 4-13
OSAM (Overflow Sequential Access Method), 3-1
redefine, 4-9
user database, 3-6
VSAM, 1-3, 1-6, 2-1, 2-2, 3-1, 3-6, 3-15
VSAMESDS, 4-9, 4-13
VSAMKSDS, 4-9, 4-13
VSAMRRDS, 4-9, 4-13
VSMHESDS, 4-9, 4-13
VSMHKSDS, 4-9, 4-13
VSMHRRDS, 4-9, 4-13
DATE command, A-2
DATE parameter, 5-26, 5-27, 5-54
dates, 4-22
convert, 4-22, 7-1, 7-3, 7-20
create, 5-44
edit, 7-20
inclusion/exclusion, 5-27
IXXUCON conversion exit, 7-20
MAPGEN statements, 4-26
MM-DD-YY, 4-22
modify, 5-44
OPTIONS statement, 5-53, 5-54
YYMMDD, 4-26
YYYY.DDD, 5-42
YYYYDDD, 5-57, 6-43
YYYYMMDD, 5-26, 5-27
DB2, 1-3, 3-12, 6-4
authorization ID, 4-14, 4-38, 6-4
DB2 option, 1-4
IBM DB2 tables, 4-64
IXXUIN input exit, 7-10
modules, D-3
plan name, 6-4, 7-4
subsystem ID, 7-4
subsystem name, 6-4
table name, 4-14, 4-38, 6-4
DB2 catalog program (DB2CATA), 4-36
DB2MAP statement, 4-36, 4-38
DB2OUT, 4-36
JCL (DB2CATL), 4-37
output, 4-39
DB2 system database, 2-4, 3-1, 6-4
access, 5-8
authorization ID, 7-4
reload, 5-24, 5-30
unload, 5-24
DB2 tables and views, 1-3
define, 4-1, 4-9
DB2 test tables and views, 3-12, 3-13, 3-14, 4-56
DB2 Text Editor work database, 2-11
DB2 text editor work database, 3-11, 6-32
DB2TXTCL, 5-9
DB2TXTIN, 5-9, 5-48
initialize, 5-6, 5-9
maintain with IFUCLEN, 5-9
DB2CATL, 4-37
DB2ELEM, 5-20
DB2IQRY, 5-38
DB2LOAD, 5-30
DB2MAP statement, 4-38
parameters, 4-38
syntax, 4-38
DB2SEG, 7-15, 7-24

DB2SQRY, 5-35
DB2STAT, 5-47
DB2TXTCL, 5-9, 5-51
DB2TXTIN, 5-6, 5-9, 5-48
DB2UNLD, 5-24
DBD (Database Description)
  specify DBD name, 4-14
DBD (database description), 3-6
DBDGENs
  secondary indices, 4-42
  variable length segments, 4-42
DC users, 6-30
DD statements
  for text editor work database, 6-31
DDF (DEFINE DIRECTORY FUNCTION), A-4
  required for IXUSQRY, 5-18
DDFEDL, 7-14
DDI (DEFINE DIRECTORY INQUIRY), A-4
  required for IXUSQRY, 5-18
  required for text editor, 5-18
default directory, 1-1
DEFER, 6-10, 6-11, A-1
DEFINE, A-1
DELETE, A-2
DELETE statement (IIGEN Utility), 4-4
DELETE statement (IIGEN utility), 4-70, 4-71
  ALL parameter, 4-70
  DIRECTORY parameter, 4-70
  LTERM parameter, 4-70
  specify in $$$$$IXX, 4-70
  syntax, 4-70
  VOCAB parameter, 4-70
device support, 1-6
  printers, 1-6, 2-2
  terminals, 1-6
directories, 2-14, 3-6, 4-3
  build, 4-1
  connected, 3-8, 4-2, 4-46
  define, 4-2, 4-44, 4-45, 4-48
  directory blocks, 3-5, 3-6
  maintain, 4-4, 4-66
  modify, 4-66
  re-create, 2-15, 5-31
  security, 4-47
  user directories, 3-7
DIRECTORY statement, 4-2, 4-3, 4-44, 4-66
  CDIR parameter, 4-46
  CTRAN parameter, 4-46
  MAP parameter, 4-45
  NAME parameter, 4-45
  SLIMIT parameter, 4-46, 4-51, 6-27
  STRAN parameter, 4-46, 6-25, 6-27
  VOCAB parameter, 4-46
DISPLAY, 6-27, 6-42, A-2
DL/I, 1-3, 3-12
  batch, 1-6, 2-5
  calls, 2-5
DLIBATCH, 5-7, 5-48, 5-53, 5-56, 6-16
documentation, 1-1, 1-7
  IBM OS PL/I Optimizing Compiler Programmer's Guide, 7-2
  IBM PL/I Optimizing Compiler Programmer Reference Guide, 7-9
  installing online books, 1-1
  Intraccess help, 1-4, 1-9, 6-37
  viewing, 1-2
  VISION:Inquiry for IMS and CICS Automatic Query Facility (AQF) User Guide, 1-8, 2-3
  VISION:Inquiry for IMS and TSO Getting Started Guide, 1-7
  VISION:Inquiry for IMS and TSO Release Summary, 1-8, 6-14
  VISION:Inquiry Messages Guide, 3-6, 5-11
  VISION:Inquiry Reference Guide, 1-9, 2-3
  VISION:Journey for Windows System Administrator's Guide, 1-9, 6-38
  VISION:Journey for Windows User's Guide, 1-9
DSA (dynamic storage area), 7-6
DSN220.DSNLOAD, 5-8
DYLI010, 6-40, D-4

DYLI020, 6-40, D-4
DYLI030, 6-40, D-4
DYLI0SS, 6-40, D-4
DYLIQ, 6-21

E

EDITMFS, 6-31
  match CUYIEPRM parameters, 6-31
EDITSQ, A-2
  required for text editor, 5-18
END statement, 4-5, 4-7, 5-27, 5-55
entry points
  alternate, 6-4
  IXSERRA, 7-5
  IXSERRB, 7-5
  IXSERRC, 7-5
  messages, 7-5
  user exits, 7-3
environments, 1-6, 2-2, 6-1
  batch, 2-2, 6-1, 6-15, D-1
  batch in BMP, 6-1, 6-15
  BMP, 1-6, 2-2, 6-1, 6-14, D-1
  DL/I batch, 6-1, 6-15, D-1
  IMS batch, 1-6
  IMS/DC, 1-6
  MPP, 2-2, 2-12, 4-54, 6-1, 6-12, D-1
  online (MPP), 6-1
  PL/I, 7-6
  TSO, 1-6, 2-2, 4-54, 6-1, 6-21, D-1
ERRHELP statement, 5-4, 5-12, 5-14
ERRLOAD statement, 4-4, 5-4, 5-12
  before APPL statement, 4-7
  before APPL/END groups, 5-12
ERRMSG statement, 4-4, 5-4, 5-12, 5-13
ESDS, 7-15, 7-24
ESDS (Entry Sequenced Data Set), 3-16
EXCLUDE statement, 4-3, 4-4, 4-47, 4-68, 5-5, 5-25, 5-26
  DATE parameter, 5-26
  MAPNAME parameter, 4-47, 4-48
  syntax, 4-47
  VOCAB parameter, 4-47, 4-48
Explorer, 1-2
EXTRACT, 6-2, 6-27, 6-29
  considerations, 6-30
  IXXEXTRA data set, 6-27, 6-30

F

FDVALN, 7-27
FDVALO, 7-27
FDVLTH, 7-27
field definitions, 4-9, 4-26
  concatenate, 4-25
  IMS (DL/I) data sets, 4-26
  IMS secondary index, 4-9
  overlap, 4-25
  redefine, 4-25
  VSAM alternate index, 4-9
  VSAM data sets, 4-26
FIELD statement, 4-17
  DBTYPE parameter, 7-20
  describe fields, 4-10
  DESCRIPT (DESC) parameter, 4-24
  from COBOL field definitions, 4-33
  IXXUCON conversion exit, 7-20, B-10
  KEY parameter, 4-22
  NAME parameter, 4-24
  OCRCNT parameter, 4-25
  OFFSET parameter, 4-23
  OUTLTH parameter, 4-22
  parameters, 4-21
  PASSWORD parameter, 4-24
  SCALE parameter, 4-22
  SUBSTR parameter, 4-21
  syntax, 4-18
  TYPE parameter, 4-21
fields, 3-6

binary, B-2
change characteristics, 7-1
character, B-2
convert, 7-1, 7-20, B-2, B-8
database maps, 3-6
dummy, 7-31
editing, 4-22
key fields, 3-6, 4-15, 4-22
limit access, 3-7
non-contiguous, 4-21
non-key fields, 3-6
non-unique key, 4-22
not displayable, B-2
numeric character, 4-21
packed decimal, 4-21
search field, 4-22
subfield, 4-21
temporary, 6-28, 7-22
TYPE=0-9, 4-21, 4-24, 7-1, 7-20, 7-22
TYPE=U, 4-21, 4-24, 7-1, 7-20, 7-22, B-8, B-10
value security, 7-1
zoned decimal, 4-21
FINISH statement, 4-5, 4-8, 4-9
foreign language support, 3-5, 3-6, 5-16
vocabularies, 3-5
free space pool, 3-7
FTS1, 6-40
FTS3, 6-40
FTSPSBC, 5-8
FTSPSBL, 5-8
function codes, A-1
function exits, B-14
add new function, B-17
bit testing with IXXUFTM, B-18
code table entries, B-17
coding considerations, B-22
dynamically loaded, B-14, B-17
invoke, B-23
IXXUFNC, 7-1, 7-25, B-14
messages, B-18
order table entries, B-17
resident, B-14, B-17
table, B-14, B-17
function value array (FDVALUE), 7-27, 7-28
function value length (FDVLTH), 7-27
number of elements (FDVALN), 7-27
offset (FDVALO), 7-27
functions, 2-15, 7-1, 7-25, 7-26, 7-27
convert, 2-16, 5-5, 5-36, 5-37
function statement parameters, 7-26
function statement syntax, 7-26
multiple, 7-25
source format for editing, 5-36
stored, 2-15, 5-36
unload, 2-15, 5-1, 5-5, 5-31, 5-34, 5-35

G

GDG (Generation Data Group), 6-30
generic LTERMs, 3-9, 4-70

H

HDAM, 1-7, 2-4, 2-12, 3-11, 6-32, 6-41
hexadecimal, B-2, B-8
HIDAM, 2-4, 3-10, 6-35

I

IAOI01, D-3
IAOINIT, 2-16, 5-1, 5-2, 5-6, 5-48, D-3
JCL (AQFINIT), 5-48
IAOMODS, 6-34
IDCAMS, 2-14, 2-16
IDFTSP7, 6-40
IFUCLEN, 2-17, 5-1, 5-6, D-2, D-4
delete download data, 5-57
JCL (DB2TXTCL), 5-9, 5-51
JCL (IMSFTSCL), 5-9, 5-57

JCL (IMSTXTCL), 5-8, 5-50
IFUINIT, 2-16, 5-1, 5-6, D-2, D-4
JCL (IMSFTSIN), 5-9, 5-53
JCL (IMSTXTIN), 5-8
OPTIONS statement, 5-2
II, 5-56, 6-1, 6-12, D-1
II (application name), 5-42, 5-56
II.CNTL
AQFINIT, 5-48
DB2CATL, 4-37
DB2ELEM, 5-7
DB2INIT, 5-7
DB2IQRY, 5-7, 5-38
DB2LOAD, 5-7, 5-30
DB2SQRY, 5-7, 5-35
DB2STAT, 5-47
DB2TXTCL, 5-9, 5-51
DB2TXTIN, 5-6, 5-9
DB2UNLD, 5-7
DYLIQ, 6-21
IMSCOBL, 4-27
IMSELEM, 5-7
IMSFTSCL, 5-57
IMSINIT, 5-7
IMSIQRY, 5-7, 5-37
IMSLOAD, 5-7
IMSSQRY, 5-7, 5-34
IMSSTAT, 5-47
IMSTXTCL, 5-50
IMSUNLD, 5-7
IQBATD, 6-17
IQBATV, 6-20
VSAMCOBL, 4-29
II.MACLIB
Add IXHEX macro, B-8
IXSECTY, B-31
IXSGEN, B-31
II.SRCLIB
AQFHnn, 6-33
AQFIMS, 6-32
AQFMnn, 6-32
AQFMnnA, 6-32
AQFPSBIN, 5-48
EDITMFS, 6-31
IIDMGEN, 4-56, 4-64
IIERROR, 5-11, 5-14
IXXUCONS, B-5
TEXTHLP, 6-31
TEXTMFS, 6-31
IIATTACH, 6-21, 6-23, D-1
IIBATCH, 6-1, 6-15, D-1
execute with DLIBATCH, 6-16
execute with IMSBATCH, 6-16
execution JCL, 6-16
in batch, 6-15
in BMP region, 6-15
in IMS (DL/I) batch, 6-15
input, 6-15
INQIMS is transaction code, 6-15
JCL (IQBATD), 6-17
JCL (IQBATV), 6-20
LTERM is BATCH, 6-15
output, 6-16
IIBMP, 6-1, 6-14, D-1
continuous mode, 6-14
conversational mode, 6-14
DB2 access, 6-14
JCL for VISION:Inquiry, 6-15
output, 6-14
IIDEMO, D-2
IIDMDIR, 4-64, 4-69
IIDMGEN, 4-56, 4-64, 6-42
IIDMMAP, 4-69
IIERROR, 5-11, 5-14
IIGEN, 2-14, 3-6, 3-9, 4-1, 4-3, 5-1, 5-7, 5-11, 6-40, D-1
APPL statement, 4-5, 4-7
coding rules, 4-6
DBDGEN with secondary index, 4-42
DBDGEN with variable length, 4-42
define alternate keys, 4-43
define applications, 4-7
define databases, 4-1, 4-9, 4-41, 4-42, 4-44

define directories, 4-1, 4-44, 4-45
define messages, 4-1, 5-4
define terminals, 4-1, 4-49, 6-40
define test systems, 4-56
define variable records, 4-43
define vocabularies, 4-1, 5-4
define VSAM hierarchical data sets, 4-44
DELETE statement, 4-4, 4-70
DIRECTORY statement, 4-2, 4-3
END statement, 4-5, 4-7
ERRHELP statement, 5-2
ERRLOAD statement, 4-4, 5-2
ERRMSG statement, 4-4, 5-2
EXCLUDE statement, 4-3, 4-4, 4-47
FINISH statement, 4-8
input, 4-64
IXXUOUT TARGET parameter, 7-14
JCL (DB2ELEM), 5-7
JCL (IMSELEM), 5-7
LTERM statement, 4-2, 4-4
maintain applications, 4-66, 4-70, 4-71
maintain database maps, 3-6, 5-1
maintain directories, 3-7, 4-4, 5-1
maintain vocabularies, 4-4, 5-1
MAPGEN statement, 4-9, 4-10
MAPGENs, 4-3, 4-41, 4-43
messages, 3-6, 4-65
MSGLIST statement, 4-4, 5-16
names, 4-5
output, 5-20
SYSLOAD statement, 4-4, 5-18
SYSTEM statement, 4-4, 5-17
transaction codes, 4-1
UPDATEDIR statement, 4-4

IIINIT, 2-14, 3-3, 3-5, 5-1, 5-2, 5-4, 5-7, 5-10, D-1
DIRECTORY statement, 5-2
INDEX statement, 5-2
INDEX/DIRECTORY statement, 5-4, 5-10
JCL (DB2INIT), 5-7
JCL (IMSINIT), 5-7, 5-10
MSGLIST statement, 5-2
SYSLOAD statement, 5-2
SYSTEM statement, 5-2

IIPSB01, 5-7

IIPSB02, 5-7

IITSO, 6-1, D-1

IIVOCAB
required for IXUIQRY, 5-18

IMS, 1-3, 1-6, 2-1, 3-12

IMS (DL/I) databases, 1-3
define, 4-1, 4-9
EXTRACT command, 6-27

IMS (DL/I) system database, 2-4, 3-1, 6-4, 6-16
access, 5-7
initialize, 5-10
PCB address, 7-4
reload, 5-24, 5-28
unload, 5-23

IMS (DL/I) test databases, 3-12, 4-56
PLANT, 4-56
SKILL, 4-56

IMS (DL/I) Text Editor work database, 2-11
maintain with IFUCLEN, 2-17

IMS (DL/I) text editor work database, 3-11, 5-1, 6-32
allocate, 5-48
IMSTXTCL, 5-8, 5-50
IMSTXTIN, 5-8, 5-48
initialize with IFUINIT, 5-1, 5-6, 5-48
PSB is FTSPSBC, 5-8
PSB is FTSPSBL, 5-8

IMS (DL/I)Text Editor work database
initialize with IFUINIT, 2-16

IMS Connect facility, 2-12

IMS TOC (IMS TCP/IP OTMA Connection), 6-12

IMS/DB, 1-3

IMS/DC, 1-6, 2-8
BMP region, 6-14
Common Service Facility, 2-7, 2-8

IMSBATCH, 5-7, 6-16

IMSCOBL, 4-27

IMSELEM, 5-20
IMSFTSCL, 5-9, 5-57
IMSFTSIN, 5-9, 5-53
IMSINIT, 5-10
IMSIQRY, 5-37
IMSLOAD, 5-28
IMSSQRY, 5-34
IMSSTAT, 5-47
IMSTXTCL, 5-8, 5-50
IMSTXTIN, 5-8, 5-48
IMSUNLD, 5-23
INCLUDE statement, 5-5, 5-25, 5-26
  DATE parameter, 5-26
INDEX/DIRECTORY statement, 5-4, 5-10
indices, 3-2
  high level index (system database), 3-2, 3-3
  item index (system database), 3-2, 3-3
INQCOBCV, 4-27, 4-29
INQEDIT, 6-1, D-2
INQIMS, 6-15
inquiries, 1-3
  batch delimiter, 6-16
  continued, 6-10
  convert, 2-14, 2-16, 5-1, 5-5, 5-36, 5-37
  deferred, 5-22, 6-10
  internal format for execution, 5-36
  modify, 7-1
  multiple, 6-15
  non-UDO, 7-13
  source format for editing, 5-36
  stored, 2-15, 7-10
  UDO, 5-36
  unload, 2-14, 2-15, 5-1, 5-5, 5-31, 5-34, 5-35
inquiry processors, 2-4, 2-5, 6-1
  application names, 6-10
  DL/I batch, 2-9
  II or MPP inquiry program, 2-4, 2-5, 2-7, 6-1, 6-12
  IIBATCH or batch inquiry program, 2-5, 6-1, 6-15, 6-16
  IIBATCH or DL/I batch program, 2-4
  IIBMP or BMP inquiry program, 2-4, 2-5, 2-8, 6-1, 6-14
  IITSO or TSO inquiry program, 2-4, 2-6, 2-10, 6-1, 6-21
  LTERMs, 6-10
  terminal names, 6-10
  transaction codes, 6-10
installing, 1-1
  Acrobat Reader, 1-1
  documentation (online books), 1-1
Intraccess, 1-4
  capabilities, 2-3
  continuous mode, 6-38
  Java-based tool, 1-6
  processing, 2-12
  support, 1-4
Intraccess option, 1-4
IOPCB, 2-12
IQBATD, 5-5, 6-17
IQBATV, 5-5, 6-20
IQIO, 5-33
IVDEMO, D-2
IXBEGIN macro, 7-6, B-8
IXHEX macro, B-8
  source, B-8, B-9, B-10
IXRETURN macro, 7-6
IXSECTY macro, B-25, B-31, C-1
  source, C-2
IXSGEN macro, B-25, B-31, C-1
  source, C-3
  table generation, C-3
IXUIQRY, 2-16, 5-1, 5-2, 5-5, 5-7, 5-36, D-1
  JCL (DB2IQRY), 5-7, 5-38
  JCL (IMSIQRY), 5-7, 5-37
  standard vocabulary required, 5-18
IXULOAD, 2-15, 5-1, 5-2, 5-4, 5-7, 5-22, 5-24, D-1
  JCL (DB2LOAD), 5-7, 5-30
  JCL (IMSLOAD), 5-7, 5-28
IXUSER macro, 7-6, 7-27

IXUSQRY, 2-15, 5-1, 5-7, 5-31, D-1
ALL statement, 5-2, 5-5
APPL statement, 5-2, 5-5
JCL (DB2SQRY), 5-7, 5-35
JCL (IMSSQRY), 5-7, 5-34
required commands, 5-18
IXUSTAT, 2-16, 5-1, 5-2, 5-5, 5-7, 5-39, D-2
Inquiry and Function report, 5-5, 5-39, 5-44
JCL (DB2STAT), 5-7, 5-47
JCL (IMSSTAT), 5-7, 5-47
system database statistics, 5-5, 5-39, 5-40
system modifications report, 5-5
IXUUNLD, 2-15, 5-1, 5-2, 5-4, 5-7, 5-22, D-2
JCL (DB2UNLD), 5-7, 5-24
JCL (IMSUNLD), 5-7, 5-23
IXXEXTRA (EXTRACT data set), 6-27
format, 6-28
IXXFUNC function exit sample
source, B-15
IXXRMODL, D-2
IXXSTABL security table, B-30
IXXUCON conversion exit, 7-1, 7-20, B-1
considerations, 7-22, B-10
convert encoded data, 7-1
database/VSAM file name, 7-24
DISPLAY, 7-22
EXIT INDICATOR, 7-21
EXTERNAL call, 7-22
FIELD statements, B-10
INTERNAL call, 7-22
invoke, B-11
IXHEX macro, B-8
process data types, 7-1
PSB, 7-21
SEGMENT NAME, 7-24
SOURCE, 7-23
SOURCE DATA DESCRIPTOR, 7-24
STACK, 7-24
TARGET, 7-24
TARGET DATA DESCRIPTOR, 7-24
IXXUCONP conversion exit sample (PL/I), B-1, B-2
IXXUCONS conversion exit sample (Assembler), B-1, B-2, B-5, B-6, B-7
IXXUFNC function exit, 7-1, 7-25, B-1, B-14
call IXXUFTM, B-19
coding considerations, B-22
EXIT INDICATOR, 7-26
FDVFLAG, 7-28
FDVLTH, 7-28
FDVOFF, 7-28
FDVVAL, 7-28
FUNCTION DESCRIPTOR, 7-26
function exit table, B-17
invoke, 7-26
PSB, 7-26
STACK, 7-26
IXXUFNC function exit sample, B-1
source, B-14, B-16
IXXUFTM bit test function, B-18
error messages, B-24
install, B-24
link edit, B-24
source, B-19
IXXUIN input exit, 7-1, 7-10, B-1, B-25
add conditional selection, 7-1
change DB2 connection, 7-1
change system database specifications, 7-1
EXIT INDICATOR, 7-11
implement, B-31
INPUT, 7-11
invoke in a query, B-29
IXSECTY macro, B-31
IXSGEN macro, B-31
LTERM, 7-12
messages, B-30
PSB, 7-11
security table IXXSTABL, B-30
set DB2 connection, 7-10
set system database specifications, 7-10
TRANSACTION, 7-12
IXXUIN input exit sample, B-1, B-25
IXXUOUT output exit, 7-1, 7-12, B-1
change field characteristics, 7-1

DATA DESCRIPTOR, 7-15
database/VSAM name, 7-15
EXIT INDICATOR, 7-14
modify/delete selected fields, 7-1
non-UDO inquiry, 7-13
PSB, 7-14
SEGMENT NAME, 7-15
SOURCE, 7-14
TARGET, 7-14
UDO Inquiry, 7-13
IXXUOUTS output exit sample, B-1
IXXUSEC security exit, 7-2, 7-28, B-1
EXIT INDICATOR, 7-30
LTERM, 7-30
PSB, 7-30
TRANSACTION, 7-30
USER ARGUMENT 1, 7-30
USER ARGUMENT 2, 7-31
IXXUSECS security exit sample, B-1
IXXUVSM VSAM exit, 7-2, 7-5
FILE NAME, 7-32
I/O AREA, 7-32
match MAPGEN, 7-2
RECORD LENGTH, 7-32

K

keys, 2-2
concatenated key, 2-2
KSDS, 7-15, 7-24
KSDS (Key Sequenced Data Set), 3-16

L

LE run-time library, 6-12
libraries
CA-Librarian, 4-28
CA-Panvalet, 4-28
LE run-time library, 6-12
load module library, D-1, D-3
PSBLIB PSB library, 4-28
SYS1.PLILIB PL/I transient library, 6-12
licensing, 1-2
licensing (international), 1-2
licensing (U. S.), 1-2
limits, 3-7, 3-10, 4-54
100 passwords, C-1
255 maps/user directory, 3-7
500 transaction codes, C-1
checkpoint call, 6-11
database calls, 4-46, 4-51, 4-54, 6-11
length of description, 4-14
maximum 15 levels of dependency, 4-10
maximum 255 segment types, 4-10
maximum segment/records per database, 5-10
number of database calls, 3-10, 4-2, 6-10, 6-25
number of logical pages, 3-10
number of pages, 4-54
number of records (sorts), 6-27
PAGE, 6-25
page length, 6-10
SLIMIT parameter, 6-25, 6-27
sort, 6-10
load modules
AQF, D-1, D-3
native VISION:Inquiry, D-1
text editor, D-1
VISION:Journey, D-1, D-3, D-4
LTERM parameter, 5-54
LTERM statement, 4-4, 4-49, 4-66, 6-27
APPL parameter, 4-53
CKPT parameter, 4-52
DEFINITION parameter, 4-52
DIRECTORY parameter, 4-50
ERRHELP parameter, 4-52
generic terminal name, 4-50
INSERTSIZE parameter, 4-52
MFS parameter, 4-52
MFSFLDLEN parameter, 4-52
NAME parameter, 4-50
PCOUT parameter, 4-53

PCPAGE parameter, 4-53
syntax, 4-49
TERMLTH parameter, 4-51
TERMWDTH parameter, 4-52
TIME parameter, 4-51, 4-52
LTERM/TERM statements, 4-49, 4-66
LTERMs, 3-3, 3-9, 4-2, 4-4, 4-49, 4-70, 6-10, 6-23, 7-30, A-3
generic, 4-4
modify, 4-67

M

macros
IXBEGIN, 7-6, B-8
IXHEX, B-8
IXRETURN, 7-6
IXSECTY, B-25, C-1
IXSGEN, B-25, C-1, C-3
IXUSER, 7-6, 7-27
SAVE, B-22
TGET, 6-21
TPUT, 6-21
MAPGEN statement, 4-3, 4-9, 4-12, 4-13
AUTHID parameter, 4-14
DBD parameter, 4-14
DBTYPE parameter, 4-13
DESCRIPT (DESC) parameter, 4-14
DSTYPE parameter, 4-43
FILE parameter, 4-14
MAP parameter, 4-13
NAME parameter, 4-14
OFFSET parameter, 4-14
PASSWORD parameter, 4-14
syntax, 4-30, 4-32
TABLENAME parameter, 4-14
MAPGENs, 4-3, 4-9, 4-10
for DB2 tables and views, 4-11, 4-13
for IMS (DL/I) databases, 4-10, 4-12, 6-42
for VSAM hierarchical data sets, 4-12, 4-13
for VSAM non-hierarchical data sets, 4-11, 4-13
match definition, 7-2, 7-31
notes, 4-25
re-enter if modified, 4-66
secondary indices, 4-41, 4-42
variable length segments, 4-42
variable records, 4-43
VSAM record to match, 7-31
maps, 3-2, 3-6
databases, 3-7
user databases, 3-6
master control record, 6-41
reset, 5-55
messages, 2-2, 2-5, 2-14, 3-5, 3-6, 4-65, 6-27
AQF, 3-6
customize, 4-4
define, 5-4
foreign language support, 3-6
from IXXUFTM, B-24
from PL/I, 7-5
from user exits, 7-5
generate, 4-4
hard-coded, 3-6, 5-11, 7-5
in IIGEN, 4-1
in system database, 7-5
IXXUIN in put exit, B-30
list, 4-4, 5-16
native VISION:Inquiry, 5-11
return codes, 7-4
severity levels, 5-20
standard, 5-11
substitutable text, 7-5
substitutable variables, 5-13
with ERRHELP text, 4-52
MFS (Message Format Service), 1-6, 6-10
/FORMAT, 6-12
AQFIMS, 6-33
IDFTSP7, 6-40
modes, 1-3, 2-2, 4-53
checkpoint, 2-2
continuous, 1-3, 4-50, 4-53
conversational, 1-3, 4-50, 4-53, 6-10, 6-11
non-checkpoint, 2-2

non-conversational, 6-10
modes of operation
batch, 1-6
BMP, 1-6
MPP, 1-6
TSO, 1-6
MP modules, 2-7
MPP (Message Processing Program), 1-6
AQF, 2-11
execution, 2-5, 2-7
II inquiry processor, 6-12
Lose control, 6-26
Sort, 6-26
MSGLIST statement, 4-4, 5-16
multiple language support, 2-2
MVS, 2-2

N

noise words, A-1

O

ODYMFS, 6-40
online processor (II), 6-12
Open Transaction Manager Access (OTMA), 2-12, 6-12
operators, A-1
OPTIONS statement, 5-5, 5-6, 5-24, 5-27, 5-51, 5-53
APPL parameter, 5-52
DATE parameter, 5-54
FMODE (functions), 5-25
IMODE (inquiries), 5-25
LTERM parameter, 5-52
syntax, 5-53
TXT-EDIT parameter, 5-52
OS sequential data set, 6-27
OTMA (Open Transaction Manager Access), 2-12, 6-12
OUTPUT
BATCH, 6-23
output, 2-1, 6-13
column format, 6-13
format line, 7-15
return to originating terminal, 2-1
return to specified device, 2-1
row format, 6-13
UDO, 6-13
OUTPUT command, 4-69
OUTPUT statement, 4-53, 4-55

P

PAGE, 6-25
page, 6-10
length, 6-10
limits, 6-10
width, 6-10
PASS (PF10), 6-31
passwords, C-1
PCB (Program Communication Block), 4-14, 7-26
PCE (PCEXTRACT), 2-12, 4-51, 4-53, 4-55, 6-40, A-2
PDD (DISPLAY DIRECTORY DATA), A-4
PDD WHOLE, 4-24
PDD WHOLE (DISPLAY DIRECTORY DATA WHOLE), A-4
PDDDS (DISPLAY DIRECTORY DESCRIPT), 4-14, 4-24
PDF (DISPLAY DIRECTORY FUNCTION), A-4
PDF (Portable Document Format), 1-1
PDL (DISPLAY DIRECTORY LTERM), A-4
PDL (DISPLAY DIRECTORY TERM), A-4
PDM (DISPLAY DIRECTORY MAP), A-4
PDV (DISPLAY DIRECTORY VOCABULARY), A-4
performance, 3-2, 5-46, 6-26
EBCDIC collating sequence, 4-25
filled up scratch pad, 5-22

monopolizing a region, 4-54
number of users, 3-2
saving directory space, 3-8
size of system data base, 3-2
sorting, 6-26
too many synonyms, 4-25
translation time, 3-2
PF keys, 6-34
PL/I, 7-2, B-1
environment, 7-6
transients, D-1
user exits, 7-2
PLANT, 3-12, 4-48, 4-56
Portable Document Format (PDF), 1-1
product licensing, 1-2
PS (DISPLAY SYSTEM), A-4
PSB (Program Specification Block)
AQFPSBIN for IAOINIT, 5-48
FTSPSBC for IFUCLEN, 5-8, 5-9
FTSPSBL for IFUINIT, 5-8, 5-9
IIPSB01, 5-7
IIPSB02, 5-7
Parameter, 4-31
PSBGENs, 4-10

R

randomizers, D-2
record length, 3-6
RECORD statement, 4-16
parameters, 4-16, 4-33
syntax, 4-16, 4-32
regions, 2-2
batch, 2-2, 4-14, 4-55, 5-8, 5-9
BMP, 2-2, 2-5, 2-8, 4-14
IMS message, 6-11
MPP, 2-5, 2-7, 2-11, 4-54, 6-11, 6-12, 6-26, 6-27
non-DL/I, 6-16
TSO, 2-2
RRDS, 7-15, 7-24
RRDS (Relative Recod Data Set), 3-16

S

SAVE macro, B-22
SAVE/PASS (PF11), 6-31
scratch pad, 3-2, 3-10, 6-10
free up, 5-22
secondary indices, 2-2, 4-9, 4-21, 4-28, 4-30, 4-31, 4-35, 4-41
KEY parameter, 4-23
security, 1-3, 2-2, 2-13, 4-2, 4-4, 4-15, 7-1, 7-10
change output characteristics, 7-1, 7-12
EXCLUDE statement, 4-4, 4-47
field values, 7-10, 7-12
IXXTABL security table, B-30
IXXUSEC for AQF, 7-2, 7-29
IXXUSEC for native VISION:Inquiry, 7-2
multiple vocabularies, 3-5
password table, C-1
passwords, 4-14, 4-24, 7-10, B-30
profiles, 3-7
restrict access, 4-2
restrict fields, 4-2, 4-3, 4-47
restrict segments, 4-3, 4-47
restrict terminals, 4-4
restrict vocabulary, 3-7, 4-3, 4-47
system database, 4-66
terminal names, 7-10
transaction codes, 7-10
user directories, 3-7
values, 7-1, 7-12
SEGM parameter, 4-30
SEGMENT statement, 4-10, 4-15, 4-17
KEY parameter, 4-15
OCRFLD parameter, 4-17
OCRNUM, 4-17
parameters, 4-15, 4-17
PARENT parameter, 4-15, 4-17
syntax, 4-15, 4-17, 4-30

segments, 2-2, 3-11, 3-15, 3-16
description, 3-11
dummy, 5-6
fixed ESDS, 3-16
fixed KSDS, 3-16
fixed occurring, 3-15
fixed RRDS, 3-16
number of occurrences for dependent segment, 4-25
parent, 4-17
record, 3-11
root, 3-11, 4-17
types, 6-35
variable ESDS, 3-16
variable KSDS, 3-16
variable occurring, 3-15
sensitivity, 4-10
site ID, 1-2
SKILL, 4-48, 4-56
SORT, 6-2, 6-25
deferred, 4-2
delete from vocabulary, 6-26
SLIMIT parameter, 4-46, 6-25
sort files, 6-25
SORTLIB, 6-25
SORTWK01, 6-25
SORTWK02, 6-25
SORTWK03, 6-25
SORTWKnn, 6-25, 6-27
SYSOUT, 6-25
sorts, 2-3, 6-2
define BMP transaction code, 4-46
limit database calls, 4-46
limits, 6-10
restrict, 6-26
SQL calls, 2-5
SQL statements, 5-48
INSERT, 5-48
storage, D-1
subfields, 4-21, 4-25
synonyms, 3-5, A-1
SYSLOAD statement, 4-4, 5-4, 5-18
system components, 2-4
inquiry processors, 2-4
system database, 2-4
utilities, 2-4
system database, 2-2, 2-4, 3-1, 3-6, 6-4
access, 5-7
backup, 2-14
contents, 3-2
data element blocks, 3-4
DB2, 5-7
define terminals, 3-9
device descriptions, 3-2
directories, 3-2
directory blocks, 3-5
high level index, 2-4, 3-2, 3-3, 3-4
IIGEN, 5-11
IIINIT utility, 3-5
IMS (DL/I), 5-7
initialize, 2-14, 5-1, 5-4, 5-10
item index, 2-4, 3-2, 3-3, 3-4
JCL for utilities, 5-7
load, 2-15, 5-1, 5-22
logical terminal descriptions, 2-4
maps, 3-2, 3-6
messages, 2-4, 3-2, 5-1, 5-11
reload, 5-4
reorg, 2-14, 4-66, 5-4
scratch pad, 2-4, 3-2, 3-10
statistics, 3-2, 5-1, 5-5, 5-39
unload, 2-15, 5-1, 5-4, 5-22
user database maps, 2-4
user directories, 2-4, 3-7
utilities, 2-2, 5-1
vocabularies, 2-4, 3-2, 3-5, 5-1, 5-11, 5-16
system modules, D-1
SYSTEM statement, 4-4, 5-4, 5-17, 5-19
SYSVOCAB, 3-5, 3-7

T

target area, 7-14, 7-16

TCP/IP (Transmission Control Protocol/Internet Protocol), 1-4

TCP/IP OTMA (Transmission Control Protocol/Internet Protocol Open Transaction Manager Access), 6-12
- connection, 6-12, 6-37

technical support, 1-3

TERM statement, 4-64
- see LTERM statement, 4-49

terminal descriptions, 2-14, 3-9, 4-1
- generic LTERMs, 3-9
- LTERMs, 3-9
- maintain, 4-66

terminals, 1-3, 2-1, 4-1
- alternate terminal, 6-14
- associate with directories, 4-1
- define, 4-49
- define MFS or non-MFS, 4-50
- dummy, 4-53, 4-56, 4-64, 6-40
- LTERM, 7-30
- names, 3-9, 7-10, 7-12, 7-28
- TSO (default terminal name), 6-23

terminology, 1-6

TERMs
- see LTERMs, A-3

test data, 3-1, 3-12
- DB2 test tables and views, 3-1, 3-12, 3-13
- define, 4-56
- IMS (DL/I) test databases, 3-1, 3-12
- VSAM test data sets, 3-1, 3-12

Text Editor, 2-11, 2-16, 3-7, 5-36
- required commands, 5-18

text editor
- EDITMFS, 6-31
- executes in MPP region, 6-31
- MFS screen formats, 6-31
- TEXTHLP, 6-31
- TEXTMFS, 6-31

Text Editor work database, 2-11, 2-14
- DB2, 2-11, 5-9
- IMS (DL/I), 2-11
- maintain with IFUCLEN, 5-1, 5-6, 5-49
- OPTIONS statement, 5-51
- utilities, 5-1

text editor work database
- DB2, 6-32
- IMS (DL/I), 6-32

TEXTHLP, 6-31

TEXTMFS, 6-31

time
- HHMMSS.T, 5-42
- HHMMSST, 6-43

TLC (Total License Care), 1-2

Total License Care (TLC), 1-2

TRANCODE system database element, 3-3, 3-5

TRANCODE/LTERM, 7-28, 7-30

transaction codes, 2-14, 3-3, 3-5, 3-9, 3-10, 4-1, 6-10, 6-12, C-1
- APPL statement, 4-7
- for BMP, 4-2, 4-7, 4-46
- for BMP when SORT is deferred, 4-2
- for connected directories, 4-2, 4-46
- for IMS, 4-5, 4-7
- for MPP, 4-7
- II (default), 6-23
- IIAQF for AQF, 6-34
- INQIMS for BMP, 6-15
- INQIMS for IMS (DL/I) batch, 6-15
- pre-set, 6-23
- security, 7-10, 7-28

transaction names, 7-12, 7-30

Transmission Control Protocol/Internet Protocol (TCP/IP), 1-4

TSO, 1-6, 6-4, 6-21
- application name, 6-4
- CLIST is DYLIQ, 6-21
- eliminate IMS/DC requirements, 6-21

execution, 2-6, 2-10
IITSO inquiry processor, 6-21
messages, 2-6
terminal name, 6-4
TGET macro, 2-10, 6-21
TPUT macro, 2-10, 6-21

U

UDO (User Defined Output), 1-3, 5-36, 6-13, 7-13
IXXUOUT exit, 7-13
UPDATEDIR statement, 4-4, 4-66, 4-67, 4-68
CDIR parameter, 4-67
CTRAN parameter, 4-67
NAME parameter, 4-67
SLIMIT parameter, 4-67
STRAN parameter, 4-67
VOCAB parameter, 4-67
user directories, 3-7
database maps, 3-7
logical terminal descriptions, 3-7
user vocabularies, 3-7
user exits, 1-3, 2-2, 2-12, 7-1, B-1
access from AQF, 7-28
Assembler considerations, 7-6
Assembler input exit skeleton, 7-8
communication via parameters, 7-4
considerations, 7-2
definition, 7-1
entry points, 7-3, 7-5
exit indicator, 7-4
in Assembler, 7-2, 7-6, B-1
in PL/I, 7-2, 7-9, B-1
IXBEGIN macro, 7-6
IXRETURN macro, 7-6
IXUSER macro, 7-6
IXXUCON conversion exit, 2-13, 7-1, 7-20
IXXUFNC function exit, 2-13, 7-1, 7-25, B-14
IXXUIN input exit, 2-13, 7-1, 7-10
IXXUIN input exit sample, B-25
IXXUOUT output exit, 2-13, 7-1, 7-12
IXXUSEC security exit, 2-13, 7-2, 7-28
IXXUVSM VSAM exit, 2-13, 7-2, 7-31
messages, 7-5
parameters, 7-4
under each exit name, B-1
using
Acrobat Reader, 1-2
utilities, 2-2, 2-14, 5-1
conventions, 5-2
IAOINIT, 2-16, 5-1, 5-6, 5-48
IFUCLEN, 2-17, 5-1, 5-6, 5-9
IFUINIT, 2-16, 5-1, 5-6, 5-9
IIGEN, 2-14, 4-1, 5-1, 5-4
IIINIT, 2-14, 5-1, 5-4
IXUIQRY, 2-16, 5-1
IXULOAD, 2-15, 5-1
IXUSQRY, 2-15, 5-1, 5-31
IXUSTAT, 2-16, 5-1
IXUUNLD, 2-15, 5-1

V

verbs, A-1
viewing documentation, 1-2
VISION:Eighty, 6-27
VISION:Excel, 2-3, 6-39
VISION:Inform, 2-3
VISION:Inquiry, 1-3, 2-3, 6-39
AQF (Automatic Query Facility), 1-6
DB2 option, 1-4
Intraccess option, 1-4
native, 1-6
nucleus, B-18
VISION:Intraccess
conversational mode, 6-38
downloading data vs. report, 6-38
online processing, 6-37
programming considerations, 6-37
VISION:Journey, 1-7
capabilities, 2-3

continuous mode, 6-11
IDFTSP7, 6-40
load modules, D-4
processing programs, 2-11
VISION:Journey download database, 1-7, 2-4, 2-12, 2-14, 3-11, 5-9, 6-2, 6-11
delete, 6-40
fields, 6-41
FTSDESC segment, 6-44
FTSRECS segment, 6-44
FTSROOT segment, 6-43
initialize with IFUINIT, 2-16, 5-1, 5-6, 5-9, 5-52
maintain with IFUCLEN, 5-1, 5-6, 5-9, 5-53
MAPGEN, 4-64, 6-43
utilities, 2-4, 5-1
VISION:Results, 2-3, 6-27, 6-39
vocabularies, 2-2, 2-4, 2-14, 3-5, 4-1, 5-16, 6-10, A-1
codes, 5-18, A-1
define, 4-46, 5-4, 5-16
foreign language support, 2-2, 3-5, 5-16
identifiers, 3-5
maintain, 4-4
multiple, 3-5, 5-16
noise words, 5-18
required commands, 5-18
security, 3-7
symbols, A-1
synonyms, 3-5, 5-18, A-1
SYSVOCAB, 3-5, 3-7
VSAM, 2-2
change records, 7-31
I/O requests, 2-5
IXXUVSM VSAM exit, 7-31
user exit, 2-13
VSAM (Virtual Storage Access Method), 3-12
VSAM data sets, 1-3
define, 4-1, 4-9
ESDS, 4-9, 4-13
hierarchical, 1-7, 3-15
KSDS, 4-9, 4-13
non-hierarchical, 1-7
RRDS, 4-9, 4-13
VSAM hierarchical data sets
define, 4-44
IXXUCON conversion exit, 7-24
IXXUOUT, 7-15
VSAM hierarchical test data sets
VSHPLANT, 4-56
VSHSKILL, 4-56
VSAM non-hierarchical data sets
COBOL converter, 4-32
ESDS, 7-15, 7-24
IXXUCON conversion exit, 7-24
IXXUOUT output exit, 7-15
KSDS, 7-15, 7-24
RRDS, 7-15, 7-24
VSAM non-hierarchical test data sets
VSPLANT, 4-56
VSSKILL, 4-56
VSAM test data sets, 3-12, 4-56
hierarchical COBOL record layout, 3-15
non-hierarchical, 3-14
VSAMCOBL, 4-29
VSHPLANT, 4-56
VSHSKILL, 4-56
VSPLANT, 3-14
VSSKILL, 3-14, 4-56

W

web page
Computer Associates, 1-3
wildcard character (*), 4-50
Windows, 1-2