Advantage™ VISION:Inquiry® for CICS®

Release Summary

6.5

Computer Associates®

IQRSC065.PDF/D07-480-010

Contents

Chapter 1: Introduction

Chapter 2: Installation and Upgrade Considerations

Chapter 3: Tips and Techniques

Chapter 4: Debugging Aids

Appendix A: PTFs and APARs

Index

1 Introduction

This book describes changes to Advantage™ VISION:Inquiry® (hereinafter referred to as VISION:Inquiry) Version 6.5.

Note: This book was previously known as the VISION:Inquiry *for CICS®* *Customer Bulletin.*

Organization of this Guide

This book is organized as follows:

Chapter 2, "Installation and Upgrade Considerations" describes installing this release. It complements installation procedures described in the *Advantage* VISION:Inquiry *for CICS Installation Guide*.

Chapter 3, "Tips and Techniques" is a collection of technical suggestions for using the system, as well as tips on how to best take advantage of VISION:Inquiry features.

Chapter 4, "Debugging Aids" contains useful debugging information and early warnings about discrepancies in the system. Organization is by topic.

Appendix A, "PTFs and APARs" contains descriptions of PTFs (Program Temporary Fixes) and APARs (Authorized Program Analysis Reports).

Version 6.5 Enhancements

VISION:Inquiry Version 6.5 contains the following major enhancements:

Partial Matching in IF Selection:

A new operator, LIKE, is added to the IF selection logic for partial matching of character strings. The "%" character is used to stand for zero or more characters in the search pattern, while the "_" character stands for exactly one character in the field's value.

New Command for Stored Queries:

A new command is introduced to display the name of the stored query, the database name, the terminal defined in the query, the date, and time on one line and the comment on the following line.

In case of no comment in the stored query, a blank line follows the stored query name line.

AQF Stored Query Panel Enhancement

The existing AQF stored query panel is changed to allow PF10 key to switch to a new panel that displays the comments associated with the stored queries as well as the other statistics. On the new panel, there are two lines for every stored query. The first line shows the stored query information as appeared in the prior releases and the comment associated with the stored query will appear on the second line. A blank line is displayed in case of no comment in the stored query.

Title in Non-UDO Queries

The Non-UDO query is enhanced to allow one or two title lines (up to 60 characters) on top of each page of the report.

LMP Support

The product code generation and support, which existed in releases prior to 6.5, is removed. VISION:Inquiry uses the Computer Associates License Management Program (LMP) in this release, which provides a standardized and automated approach to the tracking of licensed software.

SMP/E Support

VISION:Inquiry utilizes the IBM's SMP/E facilities for installation of the product.

Contacting Computer Associates

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

2 Installation and Upgrade Considerations

This chapter contains general information for VISION:Inquiry Version 6.5.

Installing Version 6.5

The instructions for installing VISION:Inquiry Version 6.5 are in the *Advantage VISION:Inquiry for CICS Installation Guide*. The following are hardware and software requirements for VISION:Inquiry and its options, as well as some installation techniques.

Hardware Requirements

VISION:Inquiry Version 6.5 can run on all IBM® and IBM-compatible enterprise servers.

Software Requirements

VISION:Inquiry operates under the following operating systems:

- OS/390® V2 R8.0 or higher
- z/OS

The operating system must also include:

- OS/390 V2R7.0 SMP/E or later.
- OS/390 V2R8.0 Language environment (LE) or later.
- OS/390 V2R8.0 TSO/E or later.
- CICS Transaction Server Version 1.3 or later.

- IMS DB Version 6.1 or later
  Required if you are using an IMS system and/or user database.
- DB2® Version 5.1 or later
  Required if you are using a DB2 system and/or user database
  (DB2 notations throughout this guide are of special interest to DB2 customers.)

VISION:Journey for Windows Requirements

The hardware and software requirements of VISION:Journey® are described in the VISION:Journey *for Windows System Administrator's Guide*.

Intraccess Requirements

See the Intraccess and HostConnect Server documentation for specific hardware and software requirements for Intraccess and HostConnect Server support.

Upgrading from Previous Versions

This section describes upgrading from previous versions of VISION:Inquiry.

Upgrading from Version 6.*x* and Version 5.*x*

The following actions are the required post-installation steps when upgrading from versions 6.2, 6.1, 6.0, 5.7, 5.6, 5.5, or 5.4.

1. Execute the current version IXUUNLD unload utility.
2. Run the Version 6.5 IIINIT utility for the system database.
   Inquiries are stored in two formats, internal and source, in this version. If you are upgrading from version 5.*x*, increase the number of directory blocks of the system database during the IIINIT run for the new source format of stored inquiries.
3. Execute the Version 6.5 IXULOAD reload utility with the unloaded data from Step 1.
4. Execute the Version 6.5 IIGEN utility with the Version 6.5 error messages and vocabulary.
5. Execute the IIGEN utility with the Version 6.5 command UPDATEDIR to update your directories with the Version 6.5 vocabulary. The vocabulary is in II.TCUYSRC (IIVOCAB). For detailed information about the UPDATEDIR command, see the *Advantage VISION:Inquiry for CICS Technical Reference Guide*.

Additional Pre-6.0 Upgrade Considerations

If you are upgrading from any version prior to version 6.0, your system may require one or more of these additional upgrade procedures.

Editing Pre-6.0 Stored Inquiries

If you are upgrading from a version prior to 6.0 and want to edit your currently stored inquiries using the Text Editor facility, do Procedure 1 or Procedure 2, as follows. Note that Procedure 1 is a two-step process.

Procedure 1

- Run the new stored inquiry and function conversion utility (IXUIQRY). It converts the stored inquiries and functions in the system database from internal to source format and saves them in a sequential data set. The *Advantage VISION:Inquiry for CICS Technical Reference Guide* contains detailed information about this utility.
- Run the Version 6.5 batch version of the product (IIBATCH) with the above sequential data set as input to store the inquiries and functions back in the system database. The stored inquiries will be in the format acceptable to the Text Editor facility.

Procedure 2

If you already have the source of the stored inquiries, Computer Associates® recommends using it as input to the batch version instead of the source that the IXUIQRY utility creates. The source inquiry and function created by IXUIQRY can appear different from the way it looked when the inquiry was stored, especially if it is a UDO (User Defined Output) inquiry.

AQF Post-Installation Upgrades

1. Regenerate the AQF BMS screens.
2. If you have changed the BMS mapset names for the AQF screens, make similar changes to the II.TCVLSRC(CVLCAMOD) member and assemble and link it, using the JCL member in II.TCVLCNTL(ASMAQFM). You must copy the generated load module to your online library.

Note: The name of the load module (CVLCAMOD) cannot be changed.

Text Editor Post-Installation Upgrades

Edit and make the changes if necessary to the Text Editor parameters in the member II.TCVLSRC(CVLCPARM) and assemble and link it, using the JCL member in II.TCVLCNTL(ASMCPARM). You must copy the generated load module to your online library.

Note: The name of the load module (CVLCPARM) cannot be changed.

3 Tips and Techniques

All references to hierarchical structures within this chapter apply to both IMS (DL/I) databases and VSAM hierarchical files.

System Database Optimization and Operations

This section discusses system database optimization (allocating space by blocks), impact of deferred inquiries on the system database space, efficiency hints for inquiries, system database integrity under TSO, connected directories, and use of the EXCLUDE statement with connected directories.

Allocating the System Database

The INDEX and DIRECTORY parameters of the IIINIT program determine how much space VISION:Inquiry will use for the system database. The space allocation for the system database can be calculated in terms of blocks instead of tracks or cylinders as follows:

- For OSAM: INDEX blocks + DIRECTORY blocks + 1
- For VSAM: INDEX blocks + DIRECTORY blocks + 2

If more than one root anchor point per block is specified in the DBD, the following formula should be used:

For OSAM:

$$\frac{\text{INDEX blocks + DIRECTORY blocks + Root Anchor Points - 1}}{\text{Root Anchor Points}} + 1$$

For VSAM:

$$\frac{\text{INDEX blocks + DIRECTORY blocks + Root Anchor Points - 1}}{\text{Root Anchor Points}} + 2$$

Round these figures up to the next whole number.

If track or cylinder allocation is required, refer to the capacity table for the device to determine the number of blocks per cylinder or track.

Deferred Inquiries and System Database Space (Checkpoint Space)

When you defer an inquiry, the information necessary to restart the inquiry is stored on the system database and given an ID number. The inquiry can be continued at any point by entering 'CONTINUE DEFERRED INQUIRY ID-number' or deleted by entering 'DELETE DEFERRED INQUIRY ID-number'.

Unless the deferred inquiry is continued or deleted, the space occupied by the checkpoint is not available for reuse. If the inquiry is never continued or deleted, the checkpoint remains until the system database is reorganized with the unload (IXUUNLD) and reload (IXULOAD) utilities.

Efficiency Hints for Inquiries

The following suggestions can improve the performance of translation and inquiry execution.

- Reduce the use of noise words to decrease translation time.
- Reduce a large user directory, if possible, to improve access in translation. Usage of synonyms in the MAPs is a typical cause of increasing the user directory.
- All names are stored in ascending sequence ($#@, A-Z, 0-9). Use earlier characters for names of more frequently referenced stored inquiries and functions, root key fields, and so on. 'ALPHA' would be a better choice for a root key than 'ZEBRA'.
- Avoid use of non-keyed segments if possible. Non-keyed segments are inefficient to process as their relative position is found by counting.

EXCLUDE Statement with Connected Directories

For the EXCLUDE statement to exclude fields from a map, that map must be defined in that directory. If the map is in a connected directory, the EXCLUDE statement will be ignored, but not flagged.

PDI and PDF Commands with Connected Directories

VISION:Inquiry can be used to access a stored inquiry or function through a directory connected to the directory which actually contains these items. However, if the stored inquiry or function needs to be displayed using PDI (DISPLAY DIRECTORY INQUIRY) or PDF (DISPLAY DIRECTORY FUNCTION) commands, this must be done through the primary directory. This is a security feature.

Secondary Indexes

This section discusses a secondary index example, generic searches on secondary indexes, multiple secondary indexes in the same inquiry, secondary indexes with duplicate keys, and PCBs for secondary indexes.

Generic Search on Secondary Index Keys

Generic keys are not supported for KEY=INDX fields. However, many secondary index key fields are composed of more than one field making generic searching desirable.

You can support this feature by defining the secondary index view as a separate map. Use the OFFSET=n parameter on the MAPGEN statement to point to the correct PCB. The root segment key is the secondary index key, or XDFLD. This field is defined with KEY=SEQ-U or KEY=SEQ-M, as appropriate. With this definition, the key can be redefined using the KEYPOS=n parameter for generic search.

Multiple Secondary Indexes in the Same Inquiry

You can specify more than one secondary index or a secondary index and primary key in the same inquiry. The data for the primary key and secondary index key(s) is taken from the PCB key feedback area. Therefore, if you display two secondary index keys in one inquiry, the same data (from the key feedback area) will be displayed for both fields. The same will occur if you display the primary key and a secondary index key.

The data in the key feedback area will depend on which sequence VISION:Inquiry uses to process the database. This is determined by the IF clause. If you want to display the data defined by several secondary index key fields (provided the data exists as one field in the segment), you must display overdefinitions of the secondary index key fields (non-key fields defined over the secondary index fields in the MAPGEN).

PCBs for Secondary Indexes

VISION:Inquiry requires that all PCBs for the same database be contiguous. When OFFSET=n is coded on a FIELD statement, VISION:Inquiry counts to the nth PCB following the first one of that DBD. If the PCBs are not contiguous, you will receive an unexpected status code from IMS. Note that OFFSET=n on the MAPGEN causes VISION:Inquiry to search for the nth PCB in the PSB.

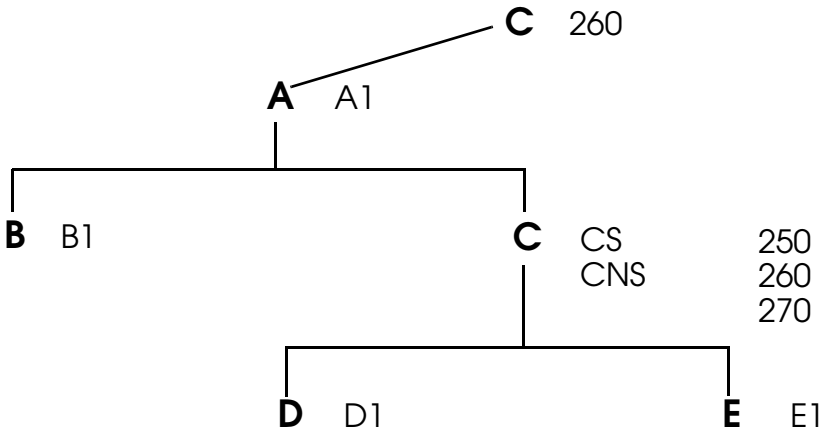
Processing in Secondary Sequence

An inquiry must specify the secondary index key field for IMS (DL/I) databases and the alternate key field for VSAM data sets in the IF clause in order to make VISION:Inquiry process in the secondary and alternate sequence.

Secondary index and alternate key fields are defined to VISION:Inquiry with the KEY=INDX parameter.

Secondary Indexing Example

Assume a secondary index database as follows:



The secondary index search field exists in segment C and is called CS. The target segment is A. A1, B1, D1, and E1 are fields in their respective segments. CNS is the secondary index search field specified in the MAPGEN as a non-secondary index field.

When you want to DISPLAY A1 CS E1 IF CS=260, you will get a report listing 250, 260, and 270. In this case, we are accessing the data on CS, the secondary index search field. Since it points to A, we get all values of CS under A.

To produce a report that prints only 260, enter the inquiry as DISPLAY A1 CS E1 IF CS=260 and CNS=260.

Secondary Index with Duplicate Key Performance Impact

Poor response time in conversational mode will occur when using a secondary index database where there is a very large number of duplicate key values compared to the call limit set.

For example, if there are 1000 duplicates and LIMIT=10, the first 10 calls are made and VISION:Inquiry must go back to occurrence 1, count to 10 to get back to where it was, then begin the next count of 10. When VISION:Inquiry goes to 20, it checkpoints and goes back to 1 again, and so on. We are not able to say "go directly to the twentieth (or twenty-first) occurrence of a secondary index value" since there is no standard DL/I command to do that. The only workaround available is to raise the call limit.

User Databases

This section discusses defining a virtual logical child segment and specifying the root key for an HDAM database.

Defining a Virtual Logical Child Segment

When defining a virtual logical child segment with "scattered" keys, you can use only one of the DBD key fields as the VISION:Inquiry key.

According to the *IMS/VS Utilities Reference Manual*, you can use any of the fields comprising the "scattered" key in an SSA. However, since all other fields must be considered, the length of the SSA field must be the concatenated length of all of the "scattered" keys. Thus, the field chosen as the VISION:Inquiry key must have a length equal to the concatenated length of all the "scattered" keys.

See the *IMS/VS Utilities Reference Manual* discussion under the SEQ operand of the FIELD statement in a DBD.

HDAM Database

The MAPGEN specification for the root key of an HDAM database is KEY=EQUAL. This parameter must be included to enable VISION:Inquiry to generate correct calls and retrieve the correct data.

Tuning DB2 Tables

This section discusses optimization techniques for DB2 tables.

DB2 Index Creation

It is highly recommended that you build indexes on the fields being used to access the DB2 tables, especially those fields specified in the IF clause of the inquiry. This will cause DB2 to use the index rather than a table space scan for accessing the tables.

DB2 RUNSTATS Utility

To improve the response time while running VISION:Inquiry against DB2 tables, execute the DB2 utility, RUNSTATS, for the DB2 tables. See the *IBM DB2 Command and Utility Reference* for additional information and usage notes.

Operational Considerations

This section describes continuing an output TERM, determining the output BMS, dynamically allocating the terminal length, and using a function user exit with character scanning.

Continuing an Output LTERM

To continue an inquiry that was sent to an output terminal, enter CONTINUE OUTPUT 'lterm name'.

Determining the Output BMS

The BMS used for output normally is the one specified by the BMS parameter on the TERM statement for the IIGEN utility. If this parameter is omitted for the TERM, VISION:Inquiry uses the mapset name specified in the MAPSET parameter of the source member CVLCPARM.

When routing the output to another terminal, you can specify the mapset name on the OUTPUT command, OUTPUT “termname” “mapset name”. This overrides the BMS= parameter on the TERM statement.

Dynamic Allocation of Terminal Length

The terminal length used for building a page of output data is calculated dynamically by VISION:Inquiry using the terminal specifications defined to CICS and the BMS map that you are using. Therefore, the TERMLTH parameter of the TERM definition at generation time is only used for UDO and the OUTPUT command. Dynamic allocation can cause some restrictions for using a footing map.

Additionally, the TERMLTH parameter is used for VISION:Journey to determine the:

- Page limit checkpoint condition when downloading data or a report.
- Placement of a heading line in the PC file when downloading a report.

Function User Exit and Character Pattern Scanning

You can perform text selection through the FUNCTION user exit. For example, you can scan FIELDD (TYPE=C, START=10, LENGTH=70) for a character pattern of 'THIS VALUE' as follows:

```
DISPLAY DB1 FIELDA FIELDB FIELDC %FUNCTION(10)=USER
(FIELDD 'THIS VALUE') IF %FUNCTION EQ 'THIS VALUE';
```

In the FUNCTION user exit, FIELDD is scanned for “THIS VALUE”. If it is found, “THIS VALUE” is returned in field %FUNCTION.

You can make this more readable through the use of an INPUT user exit as follows:

```
SCAN FIELDD FOR 'THIS VALUE' AND DISPLAY DB1 FIELDA
FIELDB FIELDC
```

In the INPUT user exit, SCAN FIELDD FOR 'THIS VALUE' AND is translated into %FUNCTION(10)=USER (FIELDD “THIS VALUE”) and the conditional selection statement is generated.

User Defined Output (UDO) Techniques

This section describes examples showing the capabilities of UDO (User Defined Output). All examples were run with the PLANT and SKILL databases supplied with VISION:Inquiry.

Logical Pages and Physical Pages

VISION:Inquiry considers the number of lines per page (TERMLTH parameter) multiplied by the number of pages (PAGE parameter) to comprise one logical page.

If your TERM specifications have TERMLTH 20 and PAGE 2, you will have a logical page of 40 lines made up of two physical pages of 20 lines each. The explicit line specification (LINE n) can have line numbers between 1 and 40, where lines 21 through 40 actually appear on the second physical page.

The UDO inquiry shown in Figure 3-1 was executed using two different TERM specifications.

```
DISPLAY PLANT FORMAT
 LINE 1 SP 20 '** LOADS-OF-FUN TOY COMPANY **'
 LINE 2 SP 5 'DATE:' DATE COL 60 'PAGE:' PAGE
 LINE 4 SP 5 'PLANT NUMBER:' PLANT.ID COL 51 'MARKETING REGION'
    PLANT.REGION
 LINE 5 SP 5 'PLANT NAME:' PLANT.NAME COL 51 'PHONE NUMBER:'
    PF(PLANT.PHONE 1 3) SP 0 '-' SP 0 PF(PLANT.PHONE 4 4)
 LINE 7 SP 10 '************************************************'
 LINE 9 SP 15 'TOY' COL 41 'PRICE' COL 51 'QUANTITY ON HAND'
 SKIP SP 7 PF(PROD.DESC 1 20) SP 3 PROD.AMT SP 2 PROD.QTY;;
```

Figure 3-1 UDO Inquiry

Figure 3-2 below shows the output using TERM with PAGE=1. For the PLANT.ID 70500, it takes two physical pages to display all the PROD segment occurrences.

```
                    ** LOADS-OF-FUN TOY COMPANY **
DATE: 12/27/02                                              PAGE:  7

PLANT NUMBER:  70500                        MARKETING REGION  SE
PLANT NAME:  DISTRIBUTION                   PHONE NUMBER:  550-7121

      ************************************************

        TOY                   PRICE          QUANTITY ON HAND
CRANE                        65.00                    88
DOLL ACCESSORIES              3.95                14,970
DOLL CLOTHES                  4.95                20,550
   .                          .                    .
   .                          .                    .
   .                          .                    .




                  ** LOADS-OF-FUN TOY COMPANY **
DATE: 12/27/02                                              PAGE:  8

PLANT NUMBER:  70500                        MARKETING REGION  SE
PLANT NAME:  DISTRIBUTION                   PHONE NUMBER:  550-7121

      ************************************************

        TOY                   PRICE          QUANTITY ON HAND
PUNCHING BAG                 29.95                   126
POST OFFICE                  29.95                    30
ROCKING HORSE                45.00                    39
ROBOTS                       18.95                   450
SLOT CARS                    38.00                   300
SKI JUMP                     58.00                    78
SKATING RINK                 28.95                    94
STEAM SHOVEL                 90.00                   135
TEDDY BEAR                   18.95                   470
TOBOGGAN RUN                 45.00                   180
```

Figure 3-2 TERM with PAGE=1

In Figure 3-3, the output is formatted using PAGE=2. For the same PLANT.ID, 70500, the second physical page is a continuation of the same logical page. The first line of the second physical page is line 21.

Because the report title and column headings are specified for lines 1 through 9, they are not repeated on the second physical page. Similarly, the page number, which counts logical pages, is not incremented.

```
                    ** LOADS-OF-FUN TOY COMPANY **
DATE: 12/27/02                                              PAGE:  7

PLANT NUMBER:  70500                        MARKETING REGION  SE
PLANT NAME:  DISTRIBUTION                   PHONE NUMBER:  550-7121

      ************************************************

        TOY                     PRICE          QUANTITY ON HAND
CRANE                          65.00                    88
DOLL ACCESSORIES                3.95                14,970
DOLL CLOTHES                    4.95                20,550
   .                            .                    .
   .                            .                    .
   .                            .                    .
POST OFFICE                    29.95                    30
ROCKING HORSE                  45.00                    39
ROBOTS                         18.95                   450
SLOT CARS                      38.00                   300
SKI JUMP                       58.00                    78
SKATING RINK                   28.95                    94
STEAM SHOVEL                   90.00                   135
TEDDY BEAR                     18.95                   470
TOBOGGAN RUN                   45.00                   180
```

Figure 3-3 LTERM with PAGE=2

Output of Literals and Data Lines

VISION:Inquiry determines whether or not to output a line or group of lines based on when data fields are to be output. Figure 3-4 shows an example in which data from the EMP segment and its dependent SAL segment are to be displayed on the same line.

- There is also a literal displayed on that line.
- A line of output is created for each occurrence of the EMP and SAL segments.
- The repeated printing of the EMP.NO field is suppressed.
- The literal 'TOTAL SALARY:' prints on every line since literals are associated with lines, not fields. To force the literal to print only when the subtotal is output, both must be specified on a different line than the detail fields.

Note also that the subtotal is calculated for each EMP.NO. Since this subtotal is associated with the EMP.NO occurrence, it is output on the same line as the EMP.NO, rather than with the last occurrence of the SAL.YTD field.

```
DISPLAY PLANT FORMAT
 LINE 1 COL 24 'LOADS-OF-FUN TOY COMPANY'
 LINE 2 COL 5 'REPORT2' COL 24 'EMPLOYEE SALARY SUMMARY' COL 55 DATE
 LINE 4 COL 9 'EMPLOYEE' COL 26 'SALARY'
 LINE 5 COL 10 'NUMBER'
 LINE LINE COL 11 EMP.NO COL 21 SAL.YTD SP 2 'TOTAL SALARY:'
     TOTAL (EMP.NO SAL.YTD) ;;


                     LOADS-OF-FUN TOY COMPANY
     REPORT2         EMPLOYEE SALARY SUMMARY        12/27/02

        EMPLOYEE        SALARY
        NUMBER

        10103       52,000.00   TOTAL SALARY:       116,000.00
                    64,000.00   TOTAL SALARY:
        10104       48,000.00   TOTAL SALARY:       107,000.00
                    59,000.00   TOTAL SALARY:
        10105       15,600.00   TOTAL SALARY:        15,600.00
        21116       15,600.00   TOTAL SALARY:        34,400.00
                    18,800.00   TOTAL SALARY:
        21124       24,000.00   TOTAL SALARY:        93,000.00
                    30,000.00   TOTAL SALARY:
```

Figure 3-4 Output of Literals and Data Lines

The inquiry in Figure 3-5 uses the PLANT.ID field as part of the report title area. Below the line with the PLANT.ID are the lines of literals comprising the column headings. These are followed by the detail field lines.

VISION:Inquiry groups literal-only lines, such as the report title and column headings, with the next database field line to determine when they should be output. Thus, the report title line is associated with the PROD segment fields. The output shows a plant with no PROD segment occurrences.

Note that the subtotal is computed for each plant, regardless of whether PROD segments exist or not. Therefore, the summary line is output for every plant. This line is under the control of the subtotal

```
DISPLAY PLANT FORMAT
 LINE 1 SP 23 'LOADS-OF-FUN TOY COMPANY'
 LINE 2 SP 27 'INVENTORY REPORT'
 LINE 4 SP 5 'PLANT:' PLANT.ID COL 56 'DATE:' DATE
 LINE 5 SP 20 '*******************************'
 LINE 7 SP 7 'TOY NAME' COL 29 'UNIT COST' COL 45 'QUANTITY' COL 66
    'INVENTORY'
 LINE 8 COL 46 'ON HAND' COL 68 'TOTAL'
 LINE 9 COL 8 '********' COL 29 '********' COL 45 '*********' COL 66
     '*********'
 SKIP SP 2 PF(PROD.DESC 1 20) SP 1 PROD.AMT SP 0 PROD.QTY
     PROD.AMT * PROD.QTY
 SKIP 2 COL 21 'TOTALS:' COL 34 TOTAL (PLANT.ID PROD.QTY) COL 56
     TOTAL (PLANT.ID PROD.AMT * PROD.QTY) ;;
```

```
                        LOADS-OF-FUN TOY COMPANY
                            INVENTORY REPORT

     PLANT:   10100                                  DATE: 12/27/02
                      *********************************

      TOY NAME              UNIT COST        QUANTITY         INVENTORY
                                             ON HAND           TOTAL
      ********              *********      ********        *********

             TOTALS:                                0            .0000
```

```
                    LOADS-OF-FUN TOY COMPANY
                        INVENTORY REPORT

     PLANT:   20150                                  DATE: 12/27/02
                      *********************************

       TOY NAME             UNIT COST       QUANTITY        INVENTORY
                                             ON HAND          TOTAL
       ********             *********       ********       *********
POST OFFICE                     29.95             30        898.5000
ROBOTS                          18.95            450      8,527.5000
SLOT CARS                       38.00            300     11,400.0000
SKI JUMP                        58.00             78      4,525.0000
SKATING RINK                    28.95             94      2,721.3000
TOBOGGAN RUN                    45.00            180      8,100.0000


              TOTALS:                          1,132     36,171.3000
```

Figure 3-5 Field as Part of Report Title Area

Forcing page breaks through the use of explicit line numbers also affects the printing of literals. Figure 3-6 below shows a simple inquiry with detail lines, subtotals, and grand totals. In this example, the grand total literals print only at the end of the report.

```
DISPLAY PLANT FORMAT
 LINE 'EMPLOYEE:' EMP.NO EMP.NAME
 SKIP 2 COL 5 'YEAR' COL 17 'SALARY'
 SKIP COL 5 SAL.YEAR COL 13 SAL.YTD
 SKIP 2 'TOTAL SALARY:' COL 19 TOTAL (EMP.NO SAL.YTD)
 SKIP 'AVERAGE SALARY:' COL 19 AVERAGE (EMP.NO SAL.YTD)
 SKIP SKIP 'GRAND TOTAL' COL 19 TOTAL SAL.YTD
    IF PLANT.ID EQ '10100' ;;
```

```
EMPLOYEE:  10103   WILLIAM AMES


     YEAR        SALARY
      94        52,000.00
      95        64,000.00


TOTAL SALARY:               116,000.00
AVERAGE SALARY:              58,000.00
```

```
EMPLOYEE:  10104      PHYLLIS LOCKMEYER

    YEAR        SALARY
     94        48,000.00
     95        59,000.00
```

```
EMPLOYEE:  10104   PHYLLIS LOCKMEYER

      YEAR         SALARY

TOTAL SALARY:                107,000.00
AVERAGE SALARY:               53,500.00
```

```
EMPLOYEE:  10105   MARY ANN THOMAS

      YEAR         SALARY
       95        15,600.00


TOTAL SALARY:                 15,600.00
AVERAGE SALARY:               15,600.00

GRAND TOTAL                  238,600.00
*
*
*
*
```

Figure 3-6 Grand Total Literal Printed at End of Report

When an explicit line number is used to force the page breaks on each occurrence of EMP.NO, as in Figure 3-7, VISION:Inquiry outputs the grand summary literals on each page. The formatting of each page is no longer controlled by the iterative printing of the EMP segment. Instead, after the subtotals are output, the remainder of the page can be formatted according to the rest of the UDO specifications. As a result, the grand summary literals are printed without the summary values.

```
DISPLAY PLANT FORMAT
 LINE 5 'EMPLOYEE:' EMP.NO EMP.NAME
 SKIP 2 COL 5 'YEAR' COL 17 'SALARY'
 SKIP COL 5 SAL.YEAR COL 13 SAL.YTD
 SKIP 2 'TOTAL SALARY:' COL 19 TOTAL (EMP.NO SAL.YTD)
 SKIP 'AVERAGE SALARY:' COL 19 AVERAGE (EMP.NO SAL.YTD)
 SKIP SKIP 'GRAND TOTAL' COL 19 TOTAL SAL.YTD
   IF PLANT.ID EQ '10100' ;;
```

```
EMPLOYEE:  10103   WILLIAM AMES


    YEAR        SALARY
     94        52,000.00
     95        64,000.00


TOTAL SALARY:               116,000.00
AVERAGE SALARY:              58,000.00

GRAND TOTAL
```

```
EMPLOYEE:  10104   PHYLLIS LOCKMEYER


    YEAR        SALARY
     94        48,000.00
     95        59,000.00


TOTAL SALARY:               107,000.00
AVERAGE SALARY:              53,500.00

GRAND TOTAL
```

```
EMPLOYEE:  10105   MARY ANN THOMAS

    YEAR        SALARY
     95        15,600.00


TOTAL SALARY:                15,600.00
AVERAGE SALARY:              15,600.00

GRAND TOTAL                 238,600.00
*
*
*
```

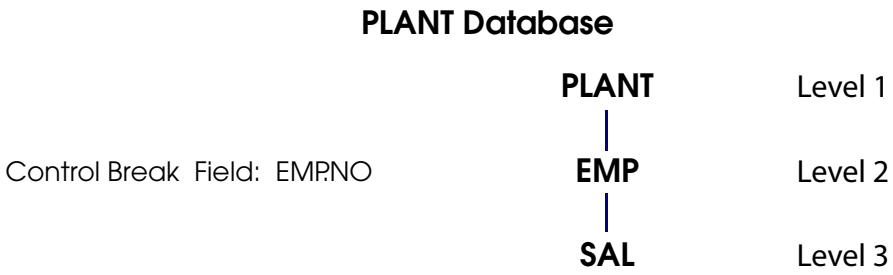
Figure 3-7 Explicit Line Number Used to Force Page Breaks

Output of Summaries

VISION:Inquiry uses the following procedure to output summaries with UDO and a subtotal calculation. VISION:Inquiry associates the summary field with the related control break field. Every time a new control break occurs, the previous summary is printed on the same line with the previous control break field.

Example:

```
DISPLAY FORMAT PLANT
     SKIP EMP.NO SAL.YTD
        TOTAL (EMP.NO SAL.YTD);;
```



Note: VISION:Inquiry collects and summarizes data in memory prior to output.

Example:

EMP. SEGMENT	10103	Total 116000	Control break
SAL. SEGMENTS	1 52000	2 64000	Two occurrences of salary data
EMP. SEGMENT	10104	Total 107000	Control break
SAL. SEGMENT	1 48000	2 59000	Two occurrences of salary data

Figure 3-8 is the output of the above inquiry:

```
10103       52,000.00                116,000.00
                64,000.00
10104       48,000.00                 107,000.00
                59,000.00
10105       15,600.00                  15,600.00
21116       15,600.00                  34,400.00
                18,800.00
21124       24,000.00                  93,000.00
                30,000.00
                39,000.00
21137       44,000.00                 150,000.00
                50,000.00
                56,000.00
21164       32,000.00                  73,000.00
                41,000.00
30201       13,400.00                  55,000.00
                19,600.00
                22,000.00
```

Figure 3-8 Output Summaries with Subtotal Calculations

When a page break occurs in the middle of lower level segment occurrences, VISION:Inquiry reprints the previous occurrence of all the higher level data in the inquiry.

According to this rule, the summary field may or may not be reprinted depending on the availability of the summary field at the page break time. This repetition also occurs in continuous mode.

The grand summary commands work the same way as the subtotal commands. The only difference is that the control break in the grand total case is associated with the highest level specified in the DISPLAY command. In other words, the total may be printed on the same line with the first occurrence of the highest level.

Output of the TOTAL Command

To improve the output format of the TOTAL command, use the SKIP or the LINE UDO commands to control the printing of the total. The following figure is an example:

```
DISPLAY PLANT FORMAT
 SKIP EMP.NO SAL.YTD
 SKIP SP 20 TOTAL (EMP.NO SAL.YTD) ;;


  10103      52,000.00
             64,000.00
                                  116,000.00
  10104      48,000.00
             59,000.00
                                  107,000.00
  10105      15,600.00
                                   15,600.00
  21116      15,600.00
             18,800.00
                                   34,400.00
  21124      24,000.00
             30,000.00
             39,000.00
                                   93,000.00
  21137      44,000.00
             50,000.00
             56,000.00
```

Figure 3-9 SKIP Command

4 Debugging Aids

To ensure the integrity and reliability of VISION:Inquiry, each new release undergoes a rigorous certification process by the Computer Associates Quality Assurance Department. However, the use of VISION:Inquiry in the numerous user environments can result in problems and limitations which are not apparent during the execution of internal and beta testing procedures.

Early Warnings

It is important to Computer Associates that our customers maintain a high level of productivity with minimal interference from product peculiarities. To that end, this chapter addresses anomalies that have been noted in the operation of the system.

VISION:Inquiry performs a syntax check on each inquiry before attempting to execute it. Due to the vast number of possible combinations for an inquiry, this syntax check may fail to detect every instance of improper usage of the language. This can result in incorrect results or ABENDs when the inquiry is executed.

If an inquiry ABENDs or produces unexpected results, verify the inquiry statement syntax with the rules in the *Advantage* VISION:Inquiry *Reference Guide*. Also verify that the map and directory used were defined correctly to the IIGEN utility. Incorrect syntax and specifications of the IIGEN input may not be diagnosed and can cause incorrect execution of inquiries.

If the inquiry syntax and map are verified to be correct, the following debugging aids should be checked to see if the problem has been previously identified. The debugging aids are categorized by the following subjects:

- System database
- Stored inquiries and functions
- Conditional selection
- Syntax
- FIND (INTER) command
- OUTPUT command

- Summaries and subtotals
- SORT command
- Operational characteristics
- Suppression and repeats
- ABENDs and loops

Use the index to find topics by symptom.

System Database

This section describes system database debugging aids and operational characteristics.

Deleting the Vocabulary

When using the IIGEN utility to delete a vocabulary, DELETE VOCAB=name must be in the application APPL NAME=$$$$$IXX. This is the application which holds the vocabularies and error messages.

Directory Names Unique Within APPL

Directory names must be unique within the APPL rather than within the entire system database.

DISPLAY DIRECTORY MAP (PDM) Command

When displaying a map from the system database using the DISPLAY DIRECTORY MAP or PDM command, only the first page is displayed.

Field and Database Name Restrictions

Once a name is specified as a database/VSAM name, it cannot be used as a field name in that map or any other map. This condition will not be flagged by IIGEN. However, unpredictable results, including messages IXX0114 and IXX0107, can occur when such a field is referenced in an inquiry.

IIINIT Parameters

The maximum number of directory blocks is 32320. Although the IIINIT utility allows a value greater than 32320 to be specified, there will be problems with the system database in the definition process if this limit is exceeded.

Key Field in a MAPGEN

If more than one field in a MAPGEN FIELD statement is defined as a key field, unpredictable results can occur during processing. The KEYPOS parameter identifies that the partially defined key is a generic key and not a redefinition of the same key.

If a key field needs synonyms, define them without the KEY parameter.

Names per MAPGEN

If you specify more than 15,000 names per MAPGEN, the IIGEN utility may ABEND with an on-error condition.

TERMLTH and TERMWDTH Parameters

The physical length of the terminal is calculated dynamically by VISION:Inquiry, except when:

- Using UDO.
- Using the OUTPUT command.
- Calculating output in horizontal or vertical format.
- Using VISION:Journey.

In these cases, calculate the correct values for the TERMLTH and TERMWDTH parameters for the terminal definition.

Stored Inquiries and Functions

This section describes debugging aids for stored inquiries and functions.

Displaying Stored Inquiries and Functions

Entering DISPLAY DIRECTORY INQUIRY or PDI with no operands lists all stored inquiries under the first map in that directory.

Entering DISPLAY DIRECTORY FUNCTION or PDF lists all stored functions under the first map in that directory.

Entering DISPLAY DIRECTORY INQUIRY WHOLE or PDIW with no operands lists all the stored inquiries for all the databases in that directory. This command does not display native SQL syntax stored inquiries.

IXUIQRY Utility and the Vocabulary

The IXUIQRY utility generates the commands and words of the standard vocabulary during the conversion of the stored inquiries and functions from internal to source in the sequential data set. The standard vocabulary is required when the sequential data set is used as input to IIBATCH to restore the inquiries and functions in the system database.

IXUSQRY Utility and the Vocabulary

The DDI and DDF commands must be present in the vocabulary when using the sequential data set created by the IXUSQRY utility as input to the IIBATCH processing program.

If you have already changed the words DDI (DEFINE DIRECTORY INQUIRY) and DDF (DEFINE DIRECTORY FUNCTION) in the vocabulary, you need to do a global change in the sequential data set (for example, using the CHANGE ALL command in ISPF edit) to substitute all the DDI and DDF commands.

Naming Stored Inquiries and Functions

Do not store an inquiry or function with the name of a field or a command. This can lead to syntax messages and unpredictable results.

Temporary Fields in Stored Inquiries

VISION:Inquiry does not support the use of substitutable values (%n) in an arithmetic variable expression in a stored inquiry. The following is an example of an invalid usage of substitutable values:

```
DDI PLANT 'INQRY' AS
DISPLAY PLANT %AMT=SAL.YTD*%1;
```

If you enter the above statement, VISION:Inquiry issues the message:

```
IXX0136 INVALID ENTRY '1' IN INQUIRY.
```

The message indicates that a syntax or semantic error was discovered while attempting to decode the above entry. Check for correct command usage and format.

Conditional Selection

This section describes debugging aids for conditional selection.

Comparing Packed and Binary Fields in DB2 Tables

Comparing a packed field with a binary field with a scale factor in an IF clause for DB2 tables will result in incorrect results and should be avoided.

Field-to-Field Comparisons

When comparing two fields from the same segment, the right hand field type is converted to the left hand field type before the comparison is made. It is, however, recommended that comparing two unlike field types be kept to a minimum because of the overhead involved.

FIRST and LAST Positional Tests

The use of the FIRST/LAST positional test on a lower level unkeyed segment can result in data being skipped or incorrect data returned. Under some circumstances, correct data can be retrieved if at least one field from all segments along the path to the unkeyed segment is displayed.

Generic Keys and OR Logic

The use of OR logic with generic keys can give incorrect results. This occurs when low order portions of the keys are used in a parenthetical Boolean expression. For example, assume that the key of the PLANT database is subdivided as follows:

```
FIELD NAME=PLANT.ID,KEY=SEQ-U,LENGTH=5,START=1,TYPE=C
FIELD NAME=PID.ONE,KEY=SEQ-U,KEYPOS=1,START=1,LENGTH=2,TYPE=C
FIELD NAME=PID.TWO,KEY=SEQ-U,KEYPOS=3,START=3,LENGTH=3,TYPE=C
```

An inquiry with the following selection returns no data if the first condition cannot be met:

```
IF PID.ONE='30' AND (PID.TWO='100' OR PID.TWO='200')
```

Correct results are obtained by changing the inquiry to:

```
IF PID.ONE='30' AND PID.TWO='100' OR PID.ONE='30' AND PID.TWO='200'
```

Key Fields (Generic and Full) and Secondary Indexing

If you use both secondary index fields and key fields (generic and full) in conditional selection, duplicate data, no data, or, in some cases, incorrect data is returned. A workaround is to add non-key fields to the MAPGEN which redefine the key fields. These non-key fields should be used with the secondary index fields in a conditional selection inquiry.

Selection on Sibling Segments

VISION:Inquiry performs data selection in hierarchical sequence. As a result, not all data for two segments on the same level is displayed if they are both used in the conditional selection.

For example, to print all occurrences of the SAL and ED segments if one of the SAL years is '93' or one of the degrees is 'MS', the following does not work:

```
DISPLAY PLANT EMP.NAME SAL.YEAR ED.DEGREE
IF SAL.YEAR='93' OR ED.DEGREE='MS';
```

Structuring the inquiry as follows gives the desired results.

```
FIND PLANT %EMP.NO = EMP.NO IF SAL.YEAR=93 OR
ED.DEGREE='MS';
DISPLAY PLANT EMP.NAME SAL.YEAR ED.DEGREE IF
EMP.NO=%EMP.NO;
```

The field EMP.NO must be a unique value for each employee segment.

Syntax

This section describes syntax debugging aids.

Batch Inquiry Processing

Batch inquiries must be terminated with two delimiters (;;). Using only one delimiter causes the batch processor to terminate without any messages.

Unidentified Characters in Inquiries (Message IXX0102)

If an unidentifiable character is detected in an inquiry, VISION:Inquiry flags the situation with a message IXX0102. This message has an insert to identify the character. The insert contains eight characters starting from the beginning of the unidentified character.

Using ';' as a Literal

Attempting to use the delimiter (;) as a literal, even if surrounded by quotation marks, causes VISION:Inquiry to fail to interpret it as a literal. The following message displays:

```
IXX0103 NO DATA DELIMITER
```

FIND (INTER) Command

This section describes debugging aids for the FIND (INTER) command.

FIND Command and AQF

AQF can only access two databases in the same inquiry (one with the FIND command and one with the DISPLAY command), such as:.

1 database	+ 1 database
1 database	+ 1 file
1 file	+ 1 file

FIND Command and Secondary Indexes

When selection in a FIND command is on a secondary index search field, subsequent DISPLAY or FIND commands may not select on that search field. Attempting to do so results in the following erroneous message:

```
IXX0600 THE INFERRED INDEX DOES NOT CONTAIN A REQUESTED FIELD
```

Changing the selection on the subsequent statements to reference an overdefinition of the search field gives correct results.

FIND Command with DISPLAY Command

When entering an inquiry using FIND with one database and DISPLAY with another, a record is not selected if the FIND command includes a reference to a lower level segment which does not exist. In this case, no connection to the second database is made.

The correct method is to use two FINDs and a DISPLAY, where the FINDs establish the relationship between the two databases and the DISPLAY references the lower level segment.

For example:

```
FIND DB1 %A=A %B=B %C=C;
DISPLAY DB2 A %B %C D IF A=%A;;
```

If %C is a field in a lower level segment which does not exist, no data is selected.

The correct way to specify this inquiry is:

```
FIND DB1 %A=A;
FIND DB2 %D=D IF A=%A;
DISPLAY DB1 A B C %D IF %A=A;;
```

OUTPUT Command

This section describes debugging aids for the OUTPUT command.

Error Messages for Routing

If errors occur during routing to another terminal using the OUTPUT command, the error messages cannot appear on the first page of the output screen. Look at all of the output pages to make sure that the output has been sent correctly.

MODE Parameter

When the OUTPUT command is used, the inquiry is processed according to the MODE parameter of the destination TERM. If an inquiry is being routed to a continuous TERM, MODE=0xx, the inquiry is processed in continuous mode. This could result in an ABEND. The MODE of all TERMs available for routing should be conversational, MODE=1xx.

Routing Using the OUTPUT Command to Another Terminal

Routing the results of an inquiry from one TERM to another through the use of the OUTPUT command can only be done if both TERMs are defined in the same directory. It does not work otherwise, even if the directories are connected, unless a security exit is provided.

Truncated Conversion User Exit Fields

If the conversion user exit expands a field to longer than the original length, the OUTLTH parameter must be coded in the map. If this parameter is omitted, the field is truncated to the column heading width in column mode and the actual field width in row mode.

Wrong BMS Mapname in OUTPUT Command

If the BMS mapname parameter is used with the OUTPUT command and the mapname is invalid, VISION:Inquiry abends and the CICS ABEND message on the screen is overridden by the VISION:Inquiry screen.

Summaries and Subtotals

This section describes debugging aids for summaries.

TOTAL Command for Numeric Fields Only

The TOTAL command is designed to function with numeric fields only. Requesting a total of a character field produces incorrect results.

TOTAL, COUNT, and AVERAGE with EXTRACT

Specifying TOTAL, COUNT, and AVERAGE with the EXTRACT command is accepted by the VISION:Inquiry syntax. However, these commands are simply ignored.

TOTAL, COUNT, and AVERAGE with PCEXTRACT

The TOTAL, COUNT, and AVERAGE commands are not executed when used in conjunction with the PCEXTRACT command in an inquiry. No error message is issued. The summary commands are simply treated as DISPLAY commands.

TOTALs, COUNTs, and AVERAGEs Lost

In conversational mode, when the logical page and the end of a database are reached at the same time, TOTALs, COUNTs, and AVERAGEs are lost except when sending the report to a PC file.

TOTALs, COUNTs, and AVERAGEs on Segments with No Occurrences

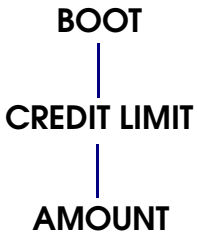
When an AVERAGE is requested for a field in a segment with no occurrences for that record, the output is '?????'. COUNT and TOTAL produce '0' for the output under the same circumstances.

Using the SUM Command with Different Segments

The SUM command can be used in arithmetic operations on fields from different segments provided they lie along a single hierarchical path. For example, SUM can be used to add up all occurrences of a field in a dependent segment and add that sum to a field in the parent.

In coding the arithmetic statement, all dependent segment fields must be to the left of the parent field.

For example:



%X=SUM(AMOUNT) + CREDIT.LIMIT is computed only once per root segment

while

%X=CREDIT.LIMIT + SUM(AMOUNT) is computed for each CREDIT.LIMIT segment.

Use of two or more SUM statements on different segments in different paths or trying to compute a SUM on an arithmetic expression involving fields from different paths produces incorrect results.

SORT Command

This section describes debugging aids for the SORT command.

Message IXX0123 with the SORT Command

If you attempt to sort a large number of fields and records in the batch version of the product, the following message may display:

```
IXX0123 STORAGE REQUIREMENTS FOR SORT EXCEEDED AVAILABLE MAIN STORAGE
```

APAR 051 has been developed to increase the available storage from 32K to 476K for IIBATCH. This APAR requires more storage to execute IIBATCH, probably using an additional 500K.

SORT with COUNT, TOTAL, and AVERAGE

Caution should be taken when using the SORT command with the subtotals (COUNT, TOTAL, and AVERAGE).

If the sort field is different from the summary control field, incorrect results could occur, as in the following example:

```
DISPLAY PLANT PLANT.ID EMP.NO EMP.NAME
COUNT (PLANT.ID EMP.NO) SORT EMP.NAME;;
```

Likewise, sorting on the field to be summarized is not only inconsistent, but produces incorrect results.

Incorrect results could also occur in the case of using the SORT command with subtotal commands (COUNT, TOTAL, and AVERAGE) applied to the fields from different legs of an IMS (DL/I) database or VSAM hierarchical data set. For example:

```
DISPLAY PLANT PLANT.ID TOTAL (PLANT.ID PROD.QTY)
COUNT (PLANT.ID EMP.NO) SORT PLANT.ID;;
```

SORT with COUNT, TOTAL, and AVERAGE on a FIND Field

Incorrect results could occur when the SORT command is used along with the subtotal commands (COUNT, TOTAL, and AVERAGE) on a field from a FIND database. The FIND database field will be considered as a constant and summarized every time a record is displayed. For example:

```
FIND SKILL %SKILL=SKILL.CODE %PLANT=PLANT.ID;

     DISPLAY PLANT PLANT.ID EMP.NAME %SKILL TOTAL(PLANT.ID %SKILL)

     IF PLANT.ID = %PLANT SORT PLANT.ID;;
```

SORT with Subtotals and SUM

The SUM command and subtotal commands (COUNT, TOTAL, and AVERAGE) are computed at different times in the data selection process. The SUM command accumulates as the data is input and its control breaks are determined by the actual hierarchical structure. Because of this, SUM may output unexpectedly when data is reordered through use of SORT. Figure 4-1 shows an example of SUM output that is not meaningful.

```
DISPLAY PLANT PROD.CODE PROD.AMT PROD.QTY PLANT.ID
        SUM(PROD.QTY) TOTAL (PROD.CODE PROD.QTY) SORT PROD.CODE ;;

PROD.CODE                PROD.AMT        PROD.QTY     PLANT.ID
                               .00               0     10100
                                                      30200

** PROD.CODE =
TOTALS                   PROD.QTY
                                0
AS                          98.00              48     40300
** PROD.CODE =AS
TOTALS                   PROD.QTY
                                48              88     50300
CC                          65.00
** PROD.CODE =CC
TOTALS                   PROD.QTY
                                88
CD                          79.00             105     40300
** PROD.CODE =CD
TOTALS                   PROD.QTY
                               105
```

Figure 4-1 Meaningless Sum Output

The SORT command, as it reorders the data, also reconstructs the hierarchy with UDO. This new hierarchy is what determines the control breaks and data suppression of the output.

When VISION:Inquiry computes subtotals (TOTAL, COUNT and AVERAGE), the control breaks are determined by the data itself. Thus, as long as the summary control fields are used as sort control fields, the summaries are correct. Use of one field as the sort control field and another as the subtotal control field can produce unexpected results.

Substring Fields, TYPE=Y

Requesting a substring as a sort field or a control grouping field gives incorrect results. VISION:Inquiry uses the entire packed field instead of the substring. However, you can assign the substring field to a temporary field and use the temporary field as a sort field or as a control grouping field in the inquiry.

Operational Characteristics

This section describes operational characteristics (conditions or restrictions that apply to selected features).

Accessing Different Legs of an IMS (DL/I) Database or VSAM Hierarchical Data Set Using PCEXTRACT

Because VISION:Inquiry performs data selection in hierarchical sequence, the data extracted by a PCEXTRACT command from different legs of a hierarchical database/data set is not in a correct format and is to be avoided.

Figure 4-2 displays a hierarchical structured database with two legs. PRODUCT is the left leg and EMPLOYEE is the right leg of the PLANT database. An inquiry using the PCEXTRACT command and requesting display of fields from both legs of a database produces incorrect results.

```
PCEXTRACT 'PROD.TXT' PLANT PLANT.ID PROD.CODE PROD.AMT EMP.NO;
```

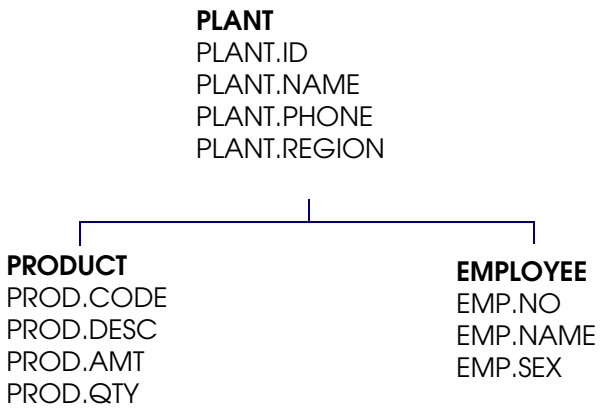


Figure 4-2 Accessing Two Legs of the PLANT Database

ASC or DSC as a Field

ASC and DSC are keywords associated with SORT. Their use as field names can cause unpredictable results such as improper SSAs being built or loops.

Column Headings Lost

Column headings may be missing on the first page following a time limit checkpoint. This occurs if there was only one line of output immediately before the checkpoint.

Displaying only FIND Temporary Fields

The display clause of a FIND/DISPLAY inquiry must have at least one database field. For example:

```
FIND SKILL %TEMP=PLANT.ID IF SKILL.NAME='SECRETARY';
DISPLAY PLANT %TEMP IF PLANT.ID=%TEMP;;
```

This results in message:

```
IXX9120 'NO QUALIFIED DATA FOUND'.
```

Division by Zero (0)

Whenever an arithmetic calculation performs division by zero, VISION:Inquiry treats this as division by one even though it is mathematically an invalid field value.

LIMIT and CONTINUE Commands

The LIMIT command is primarily intended to be used while operating in continuous mode. However, it can be used in conversational mode.

In conversational mode, whenever the page or call limit is reached and the inquiry continued, the limit on the LIMIT command is reset. For example, assume an LTERM with TIME=1 (Checkpoint after 10 calls or reads) and an inquiry with LIMIT 20. When 10 database calls or VSAM reads have been made, the VISION:Inquiry program takes a checkpoint and stops. When 'CONTINUE' is entered, VISION:Inquiry resumes with the limit counter reset to 0.

LIMIT Command and Lower Level Segments

To get a sample of the results of a query, the LIMIT command can be used. However, the LIMIT command applies to the highest level of an IMS (DL/I) database or a VSAM hierarchical data set selected (ignoring segments referred to in the selection IF criteria). You can get around this limitation by using the FIND command.

Note the differences in the following examples using the test PLANT database.

Example 1: This query will display all the employee names from the first five plants:

```
D PLANT PLANT.NAME EMP.NAME LIMIT 5;;
```

Example 2: This query will display the first five employees associated with each plant:

```
FIND PLANT %X=PLANT.ID %PLANT.NAME=PLANT.NAME; D PLANT
%PLANT.NAME EMP.NAME LIMIT 5 IF PLANT.ID =%X;;
```

Example 3: This query will display the first five employee names associated with the first two plants:

```
FIND PLANT %X=PLANT.ID %PLANT.NAME=PLANT.NAME LIMIT 2;
D PLANT %PLANT.NAME EMP.NAME LIMIT 5 IF PLANT.ID =%X;;
```

LINE and SKIP Specifications

Both the LINE and the SKIP specification can be used to control the vertical spacing of fields on a screen. Care should be exercised when using both LINE and SKIP specifications. If used incorrectly, you might get unpredictable errors such as a syntax error flagging a correct literal.

SEGMENT Statement TWIN=NO Specification

The TWIN=NO specification on the SEGMENT statement does not function properly. Use of this specification can cause unpredictable results such as duplicate data or skipped data.

Temporary Fields and Row Mode

Whenever an object variable field is the first field on a continued page in row mode and all non-temporary fields in a segment have been printed, the temporary field cannot be printed.

TERM and Conversational Mode

When VISION:Inquiry performs a page-end or call-limit checkpoint, it writes the checkpoint data on the system database by TRANCODE/TERM. If two inquiries using the same TERM and TRANCODE (and conversational mode) run simultaneously, VISION:Inquiry may confuse their checkpoints. Continuing one inquiry may actually continue from the other checkpoint.

Text Editor and the Vocabulary

The Text Editor requires that the DDI and EDITSQ commands be present in the vocabulary. However, you can define your own synonyms for these commands to be used by your users in your vocabulary.

Suppression and Repeats

This section describes suppression of field names and repeated data.

General Rules on Suppression of Repeated Data

When the SORT command is not used, VISION:Inquiry suppresses the printing of fields based on the database hierarchy. If a field from the parent is to be printed on the same line as one from a dependent, the field from the parent appears with the first occurrence of the dependent and is suppressed with subsequent occurrences.

The use of the SORT command destroys the hierarchical structure. Here, VISION:Inquiry must examine the data of the previous line printed to determine what data to suppress.

One of the features of UDO is that the entire print line is examined to determine what data is to be suppressed. This is shown in Figure 4-3. Note that ED.SCHOOL, as data from the lowest level segment, might give the incorrect impression that fewer occurrences of this segment exist than is the actual case. This is a condition under which printing “cannot be suppressed”.

The following examples in Figure 4-3, Figure 4-4, and Figure 4-5 point out the items discussed.

```
DISPLAY PLANT
EMP.NAME PLANT.ID ED.SCHOOL SORT EMP.NAME ;;

 EMP.NAME                 PLANT.ID  ED.SCHOOL
 AGNES COVINGTON          70500
 CHARLES SALTER           20150     EMORY
 DAVID YORK               60200     CORNELL
 DONALD M KING            40300     UC
                                    UC
                                    UC
 FREDERICH GRAY           30200     BOSTON U
                                    U OF MASS
 JANE LOWELL              30200
 JOAN EVANS               40300     U OF CONN
                                    MIT
                                    MIT
 JOHN HENRY CRANE         30200
 JONATHAN OAKS            50300     TEXAS TECH
 KAREN REDFERN            60200
 MADELYN BATES            50300     OBERLIN
 MARCIE MORINO            60200     TUFTS
 MARTHA WALLINGHAM        70500     OHIO STATE
 MARY ANN THOMAS          10100
```

Figure 4-3 Suppression of Repeated Data, Example 1

```
DISPLAY PLANT FORMAT
LINE 1 COL 30 'SOME SORT OF A TITLE'
LINE EMP.NAME PLANT.ID ED.SCHOOL SORT EMP.NAME ;;

                            SOME SORT OF A TITLE
 AGNES COVINGTON          70500
 CHARLES SALTER           20150  EMORY
 DAVID YORK               60200  CORNELL
 DONALD M KING            40300  UC
                                 UC
                                 UC
 FREDERICH GRAY           30200  BOSTON U
                                 U OF MASS
 JANE LOWELL
 JOAN EVANS               40300  U OF CONN
                                 MIT
                                 MIT
 JOHN HENRY CRANE         30200
 JONATHAN OAKS            50300  TEXAS TECH
 KAREN REDFERN            60200
 MADELYN BATES            50300  OBERLIN
 MARCIE MORINO            60200  TUFTS
 MARTHA WALLINGHAM        70500  OHIO STATE
 MARY ANN THOMAS          10100
```

Figure 4-4 Suppression of Repeated Data, Example 2

```
DISPLAY PLANT FORMAT
LINE 1 COL 30 `SOME SORT OF A TITLE'
LINE EMP.NAME PLANT.ID ED.SCHOOL ;;

                              SOME SORT OF A TITLE
WILLIAM AMES                  10100  TULANE
PHYLLIS LOCKMEYER                    WISCONSIN
MARY ANN THOMAS
WILMA FORD                    20150
CHARLES SALTER                       EMORY
PETER ZATKIN                         RUTGERS
                                     TEXAS
SUSAN WARE                           MICHIGAN
JOHN HENRY CRANE              30200
FREDERICH GRAY                       BOSTON U
                                     U OF MASS
MITCHELL J HOOPS                     SANTA CLAR
                                     USC
JANE LOWELL
PATRICIA BLAKELY                     ARIZONA
SHARON DALEY
DONALD M KING                 40300  UC
                                     UC
                                     UC
```

Figure 4-5 Suppression of Repeated Data, Example 3

Repeat of High Level Segment Fields on a New Page

At the top of a new page, VISION:Inquiry always repeats the higher level segment fields specified in the DISPLAY statement for clarity. If only one segment is referenced in the DISPLAY statement and subtotal is not used, nothing is repeated on the new page. However, when subtotal is used or more than one segment is used, high level segment fields are repeated.

This is illustrated in Figure 4-6.

```
DISPLAY PLANT FORMAT
 EMP.SEX EMP.NO SAL.YTD
 SKIP COL 6 TOTAL (EMP.SEX SAL.YTD) ;;

  M  30201      13,400.00
                19,600.00
                22,000.00
  M  30202      48,000.00
                59,000.00
  M  30205      76,000.00
                92,000.00
                98,000.00
                 428,000.00
  F  30207      22,000.00
                26,000.00
  F  30211      30,000.00
  F  30215      17,000.00
                  95,000.00
  M  40304      58,000.00
                66,000.00
                75,000.00
                 199,000.00
  F  40306      64,000.00
                73,200.00
```

Figure 4-6 Repetition of Higher-Level Segment Fields on New Page

In this case, the repetition of high level segment fields (EMP.NO, EMP.SEX) clarifies the output for each new page. However, VISION:Inquiry does not currently provide a specification to control repeated output of individual fields within higher level segments. Therefore, all fields from the higher level segments are displayed.

In the following case shown in Figure 4-7, the displayed fields, the control break field, and the total fields are all in the same segment (EMP). VISION:Inquiry uses the same rule to output the data. Because the control break field and total field are in the same segment level with the display fields, the output listing is confusing.

```
DISPLAY FORMAT PLANT EMP.SEX EMP.NO
SKIP COL 10 TOTAL (EMP.SEX EMP.NO);;

   M  10103
             000000000010103
   F  10104
   F  10105
   F  21116
             000000000041325
   M  21124
   M  21137
             000000000042261

   .  .              .
   .  .              .
   .  .              .
   M  40304
             000000000040304
   F  40306
             000000000040306
   F  50304
             000000000050304
   F  50322
   F  50323
             000000000100645
   M  60205
   M  60209
             000000000120414
   F  60251
   F  60258
             000000000120509
```

Figure 4-7 Control Break Field and Total Field in Same Segment Level

Suppression of Field Names in Row Mode

VISION:Inquiry suppresses output of field names in row mode when SORT is used if the field name is blank in the sorted record. VISION:Inquiry uses the blank value in sort records to indicate the absence of a field. Therefore, any field that actually exists but contains a value of all blanks is misinterpreted and the field is skipped.

ABENDs and Loops

This section describes debugging aids for ABENDs and loops.

ABENDs After Modifying a Map

If a map is changed, replace all prestored inquiries referencing that map and recreate all directories. Otherwise, executing a stored inquiry results in a loop or ABEND.

DB2 SQLCODE - 802

This error indicates that DB2 cannot process your VISION:Inquiry statement; try executing it in native SQL.

If you still get this error message, there is an SQL restriction and you need to reword or simplify your inquiry.

If this error message is received using VISION:Inquiry, but not when executed in native SQL, contact Computer Associates Technical Support for assistance.

(GE GR) 06 Messages from IXULOAD Reload Utility

If the IXULOAD reload utility terminates normally, but contains the following message, there is insufficient space on the system database:

```
(GE GR)06 — TRYING TO GET LTERM SCRATCHPAD AREA RECORD (nnn)
```

The system database should be initialized with more directory space in order to be reloaded.

Locating the Inquiry in a Dump

When an ABEND occurs, VISION:Inquiry saves the source of the inquiry in acquired storage. You can usually see the inquiry at the end of the user subpool storage in the dump.

On-Code 9050 in IIBATCH with Sorting

On-code 9050 messages appear under the following condition:

The installation standard JCL for the SORT program (SORTLIB, SYSOUT, and SORTWKnn DD statements) must be included in the batch JCL in order to successfully sort data. If this JCL is missing, IIBATCH terminates with an on-code 9050 when a sort is requested.

SC03 and other ABENDs in IIBATCH

Insufficient storage to process the input inquiry can cause SC03 or other ABENDs when running IIBATCH. Increasing the region size should remove the ABEND and allow the inquiry to be processed correctly.

S0C4 ABEND in IXXBTRN in IIBATCH

If the batch run that builds a stored inquiry ABENDs with a S0C4 in module IXXBTRN, the region size should be increased. A region of 800K is probably adequate for storing most inquiries.

User 16 (U16) ABEND with SORT

When VISION:Inquiry detects a condition which requires it to terminate when sorting (such as when the SLIMIT is exceeded), the program terminates with a condition code of 16. SORT can be installed to issue an ABEND with a user code of 16. If this is the case, the message from VISION:Inquiry explaining the reason for terminating (such as IXX0437 SORT LIMIT EXCEEDED) is lost. Changing the way SORT is installed to not ABEND allows the message to be output.

User 16 (U16) when Using SORT

If a User 16 occurs when the SORT command is used in an inquiry in the batch environment, it generally means that the SORT routine ABENDed. VISION:Inquiry traps the message from the SORT routine and produces a User 16.

An APAR has been generated to be applied to the system by SMP/E. This APAR allows the sort message to print. This message can then be researched in the proper sort manual and the problem corrected. Contact Computer Associates Technical Support for the APAR number (see Contacting Computer Associates on page 1-3).

Once the SORT routine is correctly set up, use the RESTORE command of SMP/E to remove the APAR.

User 240 (U240) or System 322 (S332) ABEND with SORT

A User 240 or System 322 ABEND displays under the following conditions:

- Incorrect placement of the ASC or DSC keywords in a SORT command can cause VISION:Inquiry to loop or ABEND. The ASC or DSC keyword must follow the SORT command, and never precede it.
- If a map is changed, all prestored inquiries referencing that map should be replaced and directories re-created. Otherwise, executing a stored inquiry could result in a loop or ABEND.

User 4000 (U4000) from IIGEN

If the IIGEN utility terminates with a User 4000, the problem can usually be traced to the input statements. For example, coding 'DBD,dbdname' instead of 'DBD=dbdname' can cause the ABEND. Not all of these conditions produce a syntax error message. Check that input statements are syntactically correct.

VISION:Inquiry Problem Reporting

In general, any problem that you suspect to be within VISION:Inquiry should be reported to Computer Associates Technical Support. In order to assist you in resolving problems in a timely manner, prior to contacting Technical Support, ensure that the following information is gathered:

- Your name, company name, telephone number, and FAX number
- Release numbers of the following products:
  - VISION:Inquiry
  - Operating System
  - DB2 or IMS, as applicable
- Documentation of the problem, such as:
  - Exact inquiry command
  - Objective of your task before the error occurred
  - The name, identification, or description of the last valid action you took before the problem occurred
  - Any error messages that were displayed
  - Steps necessary to reproduce the problem
  - Copy of the appropriate database maps
  - All pertinent PSBs and DBDs
  - The list of PTFs and APARs applied to your VISION:Inquiry system.
  - Is this a new query or one that has run successfully before?
  - Urgency of problem
  - Any other information that you feel may be helpful in resolving the problem

The Technical Support representative can also help you determine additional information or specific software documentation, such as memory dumps, so that Computer Associates can determine the cause of the problem.

Note the name of the Technical Support representative so that inquiries or accompanying documentation are directed to the proper Technical Support representative.

Problem Processing

Problems are usually processed for the *most current release* of the product. Computer Associates may not be able to help you resolve your problem if products are not the most current release or if the problem is caused by a user-written exit or Supplied User Option (SUO). Your company may be billed for time and materials if any or all of these conditions occur.

- Where necessary, the Technical Support representative who received the problem will work with the Product Development/Maintenance staff for resolution.
- Computer Associates will attempt to resolve the problem as soon as possible, depending upon the complexity of the problem, its urgency, and the release number of the product.

Supplied User Options (SUO) Procedures

From time to time, Computer Associates provides specialized enhancements or routines called Supplied User Options (SUO). They are usually delivered in source code without warranty. The availability of source code permits the user to customize the Supplied User Options. Computer Associates, of course, cannot guarantee the integrity of the SUO under these circumstances.

If the problem involves user-written code or a Supplied User Option (SUO), Computer Associates will not attempt to solve the problem unless you send written authorization to charge time and materials expenses. You may want to stipulate a maximum charge.

A PTFs and APARs

Prior to VISION:Inquiry Version 6.5, PTFs were known as SMs or GSMs (General System Modifications). These types of patches apply to all systems and correct or enhance the software system. SMP/E APAR is a customization to the system that satisfies a unique site requirement. Prior to Version 6.5, these patches were known as RSMs (Restricted System Modifications).

In this version, the PTFs and APARs are identified using the following format:

IQnnnnn

where:

IQ	indicates the VISION:Inquiry engine	
nnnnn	is the Modification Number Identifier:	
	00051 to 00099	Numbers assigned to APARs, special patches
	00101 to 00500	Numbers assigned to PTFs, general patches

The APAR control statements contain comments for each item that describes the situation addressed by the APAR. Review the description of any APAR you are considering for your system. Contact Computer Associates Technical Support if you have any questions, concerns, or if you just need more information regarding an APAR (see Contacting Computer Associates on page 1-3).

All PTFs and APARs are installed to VISION:Inquiry and its components under the control of SMP/E. The SMP/E process for handling PTFs and APARs has the following basic steps:

1. Record and save the PTF or APAR into the global zone using the RECEIVE command.
2. Use the APPLY command to install the PTF or APAR to the target libraries.
3. Use the ACCEPT command to install the PTF or APAR into the distribution libraries.

The *Advantage* VISION:Inquiry *for CICS Installation Guide* has more detailed information about the installation of PTFs and APARs.

Examples of Common APARS

The following are some of the common APARs. The source of these APARs can also be found in the II.PREP.CNTL library where the APARs are downloaded during installation of the product.

```
++APAR(IQ00051)
 /* When performing a sort for a database with a large number of
    records, the maximum VISION:Inquiry stack size of 32K is
    insufficient.  This APAR increases the maximum stack size to 476K
    for a BATCH environment.
 */
 .
++VER(Z038) FMID(CCVL650).
++ZAP(IXBPOPT).
  NAME IIBATCH IIPLIOPT
  VER 001C 00008000
  REP 001C 00077000
```

Figure A-1 APAR051: Increase the Maximum Stack Size from 32K to 476K

```
++APAR(IQ00054)
 /* The maximum length for character fields defined to VISION:Inquiry is
    255. The IIGEN utility issues an error message for character fields
    with a length greater than 255 bytes. This APAR bypasses the error
    message and allows the fields with length greater than 255 bytes to
    be defined to VISION:Inquiry; however, processing such a field in
    VISION:Inquiry produces incorrect results.
 */
 .
++VER(Z038) FMID(CCVL650).
++ZAP(IXGMAPF).
 NAME IIGEN IXGMAPF1
 VER 2222 47D021B0
 REP 2222 47F021B0
```

Figure A-2 APAR054: Bypassing Error Message for Character Fields Greater Than 255 Bytes

```
++APAR(IQ00055)
 /* This APAR changes the format of the UDO DATE and DATEF
    parameters from mm/dd/yy and mm/dd/yyyy to dd/mm/yy and dd/mm/yyyy
    respectively.
 */
 .
++VER(Z038) FMID(CCVL650).
++ZAP(IXXFOUT).
 NAME IXXFOUT1
 VER 07D0 D201E000D238
 VER 07DA D201E003D23A
 REP 07D0 D201E000D23A
 REP 07DA D201E003D238
```

Figure A-3 APAR055: Change the UDO Date to European Format

```
++APAR(IQ00056)
 /* THIS APAR IS REQUIRED IF YOUR VISION:INQUIRY TRANSACTIONS ARE TO
     BE SUPPORTED BY PARALLEL PROCESSING for a VSAM system data base,
    (RUNNING IN BOTH THE BATCH AND CICS REGIONS).  It uses the ENQ and
    DEQ macros for the VSAM system data base.
 */
 .
++VER(Z038) FMID(CCVL650).
++ZAP(IUXTVSM).
  NAME IIBATCH IXXTVSM1
  VER 0312 47F0232A
  VER 038C 47F023A2
  REP 0312 07000700
  REP 038C 07000700
++ZAP(IUCTVSM).
  NAME IXXSTRT IXXTVSM1
  VER 0124 47F0916E
  VER 01BA 47F09276
  REP 0124 07000700
  REP 01BA 07000700
```

Figure A-4 APAR056: For parallel processing using VSAM system data base

```
++PTF(IQ00101)
  /* THIS PTF APPLIES ONLY TO THE SYSTEMS WITH THE
     VISION:JOURNEY FOR WINDOWS FEATURE.
     IT RESOLVES A CICS AEYD ABEND AND COMPLETES THE REQUIREMENTS
     FOR MAKING THE VISION:JOURNEY FOR WINDOWS PROGRAM, DYLC010,
     COMPLETELY REENTRANT.
  */.
++VER(Z038) FMID(CCVL650).
++ZAP(DYLC010).
NAME DYLC010 DYLC010
VER 026E C26E,C270
VER 0272 C272,C274,C276
VER 0278 C278
VER 03DC 92FF,D7DA
VER 17B2 C052
VER 17E0 C052
VER 180E C052
REP 026E 92FF,D7DA
REP 0272 D201,D608,C052
REP 0278 07FE
REP 03DC 45E0,C26E
REP 17B2 D608
REP 17E0 D608
REP 180E D608
```

Figure A-5 PTF101: Makes the VISION:Journey routine reentrant

Index

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

O

P

R

S

T

U

V

W