Advantage™ VISION:Inform®

System Administrator Guide

4.0

Computer Associates®

IFADM040.PDF/D39-008-011

Contents

Chapter 1: The VISION:Inform System

Chapter 2: Using VISION:Inform

Chapter 3: VISION:Inform System Security

Chapter 4: Controlling Processing

Chapter 5: System Administration

Chapter 6: Using the Batch Simulator

Index

Chapter

1

The VISION:Inform System

The Computer Associates® VISION:Inform® System Administrator Guide explains how to:

- Prepare the VISION:Inform system for processing.
- Tune it for optimal performance.
- Define and maintain user profiles.
- Establish necessary levels of security.
- Control query processing by setting up the Background Processor.
- Control and monitor resource usage.
- Optimize database access and query execution.
- Use the Batch Simulator.

VISION:Inform Architecture

VISION:Inform is the host-based server in the Computer Associates client/server architecture.

VISION:Inform is supported by the following set of clients:

VISION:Bridge™

The VISION:Inform subsystem host 3270 client for administration, query preparation, processing, data downloading, and routing.

VISION:Journey® for Windows® and VISION:Journey for DOS

The independent PC clients for query preparation, processing and data downloading.

See Figure 1-1 for an overview of the basic client/server environment.

VISION:Inform Client/Server Environment

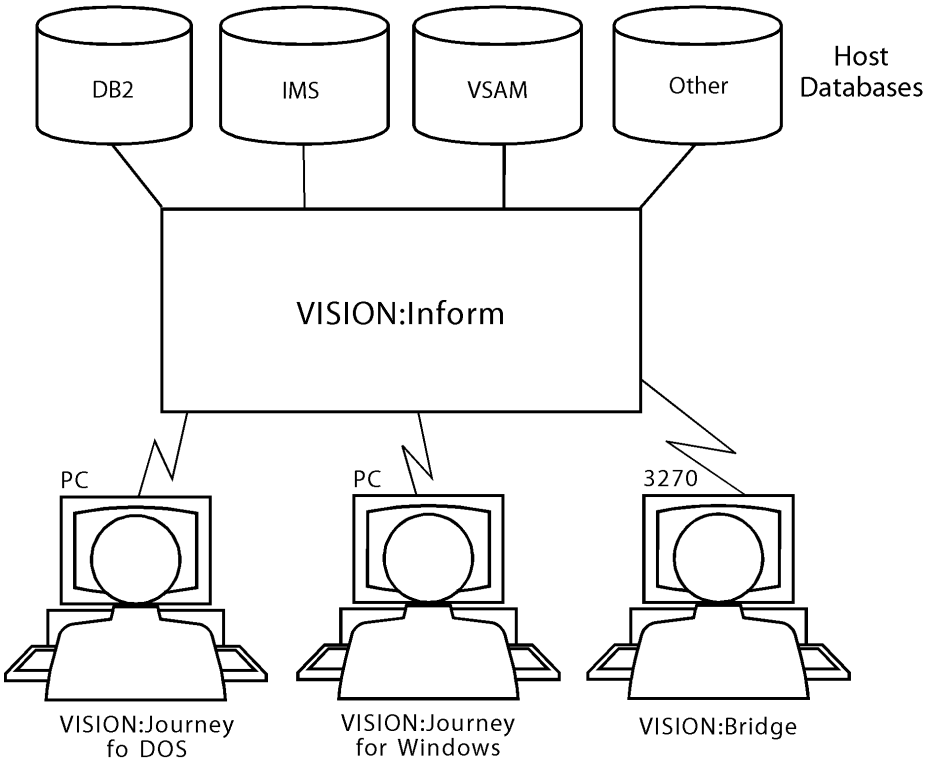


Figure 1-1 The Client/Server Environment

VISION:Inform Components

VISION:Inform and the client software products work together to provide workstation users with a cooperative processing facility to access virtually any file or data management system supported on IBM® hosts.

With the client software products, workstation users can request VISION:Inform to extract selected subsets of data, optionally summarize them, and download them to the workstation to use with other applications.

VISION:Inform is made up of a number of integrated components. Your installation administrators set up these components.

The primary components of VISION:Inform are:

- Foreground Processor
- Background Processor
- Definition Processor
- Communication File
- Foreground Library
- Background Library
- Definition Library

Typically, the system administrator installs the product, creates the online VISION:Inform profiles, defines security access to other profiles and data files, and creates queries as needed.

The database administrator usually sets up the Definition Processor, runs certain utilities, and populates the definition library, background library and the foreground library.

The host-based 3270 end users and client-based end users create and submit queries, and download the data output from those query requests.

The relationships among these components are shown in Figure 1-2.

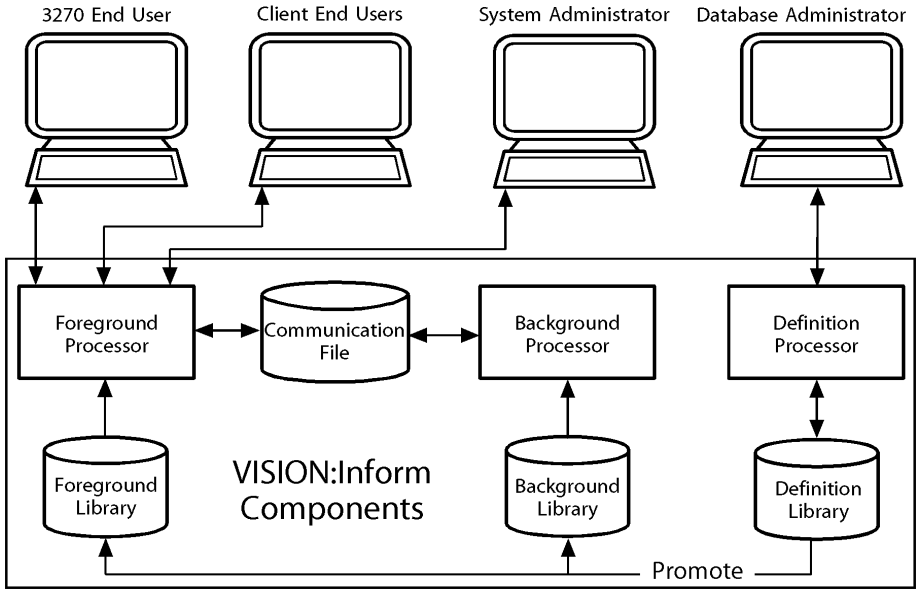


Figure 1-2 VISION:Inform Architecture

The following sections describe each VISION:Inform component.

Foreground Processor

The Foreground Processor provides two major functions:

- It provides the system administrator with facilities to define security constraints and monitor and control resource usage.
- It also acts as a communications interface to the workstation to enforce security constraints and submits queries for execution.

To the online environment in which it operates, the Foreground Processor is just another application program. It conforms to the same design constraints and operating considerations of other application programs running in that environment.

Users make requests for data, called queries or tasks, to the Foreground Processor through the client software product. The system administrator sets up security procedures using the Foreground Processor. Throughout these functions the Foreground Processor interactively checks for internal consistency. As a security measure, the Foreground Processor checks the profile to ensure that the user can access the data requested or perform the action requested.

The Foreground Processor writes a submitted query or task to the communication file where it remains until a Background Processor, for its processing class and data view, becomes active.

Background Processor

Note: VISION:Inform *uses the term "Background Processor", which may have been referred to as "Batch Processor" in previous releases*.

The Background Processor operates in the batch processing areas of your operating system. It contains a program which receives submitted client requests for processing, batches them, if possible, with other requests, and then processes them either individually or in batches.

The Background Processor retrieves the data and manipulates it according to the request's specifications.

The Background processor processes queries and tasks in the communication file by class within database sequence.

Upon completion of processing, the Background Processor returns output to the communication file. The output remains in the communication file until the client requests delivery through the Foreground Processor.

Definition Processor

The Definition Processor provides a way to develop and maintain your VISION:Inform definitions. VISION:Inform definitions include:

- Table definitions
- File definitions
- Logical data view definitions
- Procedure definitions

The Definition Processor operates in the user friendly ISPF program development environment. In this interactive environment, you can develop your definitions quickly and easily.

The Definition Processor validates all VISION:Inform definitions and then saves them in the definition library. The definition library is a partitioned data set, which holds all of your VISION:Inform definitions.

The Promote process compiles selected definitions in the definition library and copies them to the foreground and background libraries for use by VISION:Inform.

For additional information on the Definition Processor, see the *VISION:Inform Definition Processor Reference Guide* and *VISION:Inform Installation Guide* for your environment.

Foreground Library

The foreground library contains the definitions and user profiles used by the Foreground and Background Processors. It also contains stored queries created by VISION:Inform users.

- The definitions enable the Background Processor to recognize and access databases and files. When you execute the Promote Process Utility, the Promote process promotes (compiles and migrates) definitions from the definition library to the foreground library.
- User profiles provide the system security.

Background Library

The background library contains all of the definitions used by the Background Processor. The Promote process promotes definitions from the definition library to the background library.

Definition Library

The definition library is a standard, open-architecture, partitioned data set used by the Definition Processor. It contains all of your VISION:Inform definitions. When the definitions are ready to be put into production, the Promote process promotes them to the foreground and background libraries.

Communication File

The Foreground Processor and the Background Processor use the communication file to transmit information between them. Queries and tasks submitted by the client software are stored in the communication file and, if possible, batched for processing.

The communication file provides:

- A means for queries and tasks to be transferred from the workstation to the Background Processor.
- A storage medium for queries and tasks to await processing and extracted data to await delivery.

Auxiliary Components

In addition to the primary components, there are auxiliary components.

Utility Library

The utility library contains the definitions needed to run certain of the VISION:Inform system utilities.

Work Files

One internal work file is used by the Foreground Processor, and three internal work files are used by the Background Processor.

Log Files

Each Background Processor can optionally build log files every time it executes. The log files contain information about the Background Processor, the queries and tasks it executes, the databases it processes, and any problems encountered during processing. This information is available online, and in batch if the sequential log file is saved.

The VISION:Inform system administrator defines what information is written to the log files.

A sample file definition for the log files is provided with VISION:Inform.

IMS System Flow

This section describes how VISION:Inform operates and uses its components in the IMS™ environment. The next section describes VISION:Inform in the CICS environment.

When you submit a query or task under IMS (IMS/DC and IMS/TM), the query or task input is placed in an IMS message queue. The IMS control region schedules your input for submission to the Foreground Processor, which operates in the IMS message processing region. The region controller activates the Foreground Processor.

The Foreground Processor reads your query or task input from the message queue, translates it, and interactively checks for syntax and consistency. The Foreground Processor uses the foreground library to validate the query or task against file definitions and logical data view definitions. If it detects a problem, a diagnostic message displays on your terminal or workstation.

During this interactive processing, the Foreground Processor uses the work database to store or retrieve query and task statements. This is a transparent activity that takes place for various reasons. For instance, for queries or tasks larger than one screen, the work database is used to store the statements that do not fit on the screen.

VISION:Bridge (IMS)

With the VISION:Bridge component of VISION:Inform, 3270 terminal users can retrieve and manipulate data, format easy-to-read reports, and create sublevels. To accomplish this, it uses the Foreground Processor to work with queries built through either Option 4 (Quick Query) or Option 6 (Standard Query Processing) from the Main Menu and one or more Background Processors.

Figure 1-3 illustrates VISION:Bridge and the interaction of its components within IMS (IMS/DC and IMS/TM).

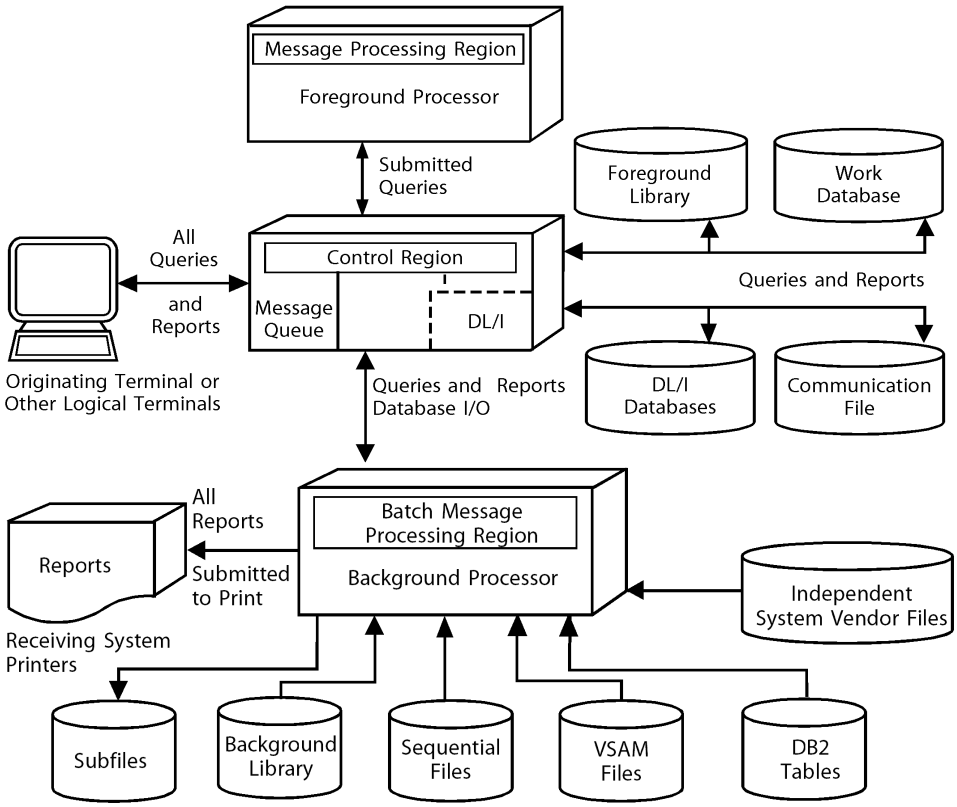


Figure 1-3 VISION:Bridge (IMS)

The communication file is the medium used to transfer queries and reports between the online and batch regions. A Background Processor, appropriate for the query class, retrieves submitted queries from the communication file and processes them.

Background Processors operate in a BMP or batch region. Figure 1-3 shows one Background Processor, but you can have one or more Background Processors operating in several BMP or batch regions.

Foreground Processor Actions

The Foreground Processor:

- Uses the foreground library, which contains descriptions of the files or databases and the user security profiles, to perform the decoding and syntax checking.
- Issues DL/I calls to access the foreground library and the work database. DL/I, residing in the control region, handles these calls.
- Writes queries to the communication file where they are held until a Background Processor becomes active.

Background Processor Actions

The Background Processor:

- Translates your query and batches it, if possible, with other queries using the same database or file.
- Uses the background library to access the definitions of fields used during processing.
- When the Background Processor runs in a BMP and issues DL/I calls to access the data stored in the user databases, the foreground library, the work database, and the communication file, DL/I residing in the control region handles the calls.

  For data requested from non-DL/I files, the Background Processor issues its own I/O instructions.
- Retrieves the requested data and manipulates it according to the specifications in the query. If the query requests that the data be sorted, the Background Processor calls the sort program, which sorts the data.
- Stores the output generated by a query in the communication file where you can request to view it at your terminal, route it to other terminals and printers, or purge it.
- Writes subfiles to data sets as they are created.

Background Processor Activation

When a Background Processor for the particular class and data view becomes active, in either the BMP or a batch region, it processes the submitted queries.

- Background Processors can become active when there are queries to process.
- Background Processors can also remain active and check the communication file at predefined intervals for queries awaiting processing.
- Background Processors remain active to process multiple batches of queries until the maximum time has elapsed (as specified in the Background Processor parameters) or the TERM command is issued. The TERM command is only available to the SYSTEM user ID.

Immediate Response (IMS)

The Foreground Processor/Immediate Response Processor retrieves and displays data interactively, without the need of a Background Processor.

Notes:

- *"Immediate Response" was previously referred to as "Inquiry."*
- *See the Main Menu in* Figure 2-2.

You build Immediate Response queries by selecting either Option 5 (Quick Query Immediate Response) or Option 7 (Immediate Response Query Processing) on the Main Menu. Figure 1-4 illustrates Immediate Response and the interaction of its components within IMS.

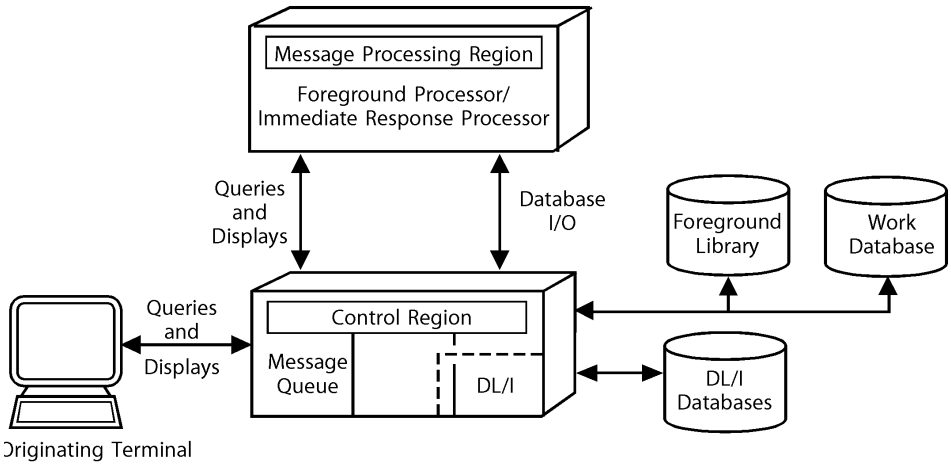


Figure 1-4 Immediate Response (IMS)

Foreground Processor/Immediate Response Processor

The Foreground Processor/Immediate Response Processor:

- Processes complete queries, DISPLAY statements, and Summary statements in the Immediate Response mode.
- Uses the foreground library to access the definitions of fields used in processing.
- Issues DL/I calls to access data stored in user databases, the foreground library, and the work database. DL/I, residing in the control region, handles these calls.
- Terminates processing of the query if it encounters problems retrieving data. The query statements display on your terminal with the appropriate diagnostic messages.
- Retrieves data only from DL/I databases.
- Retrieves the requested data and manipulates it according to the specifications of the query. If the query requests that the data be sorted, the processor calls the sort program, which sorts the data.
- Immediately returns the output generated by complete queries or DISPLAY and Summary statements to the originating terminal by way of the message queue.

  In Immediate Response, you cannot route output to other terminals. However, you can print hard copy output by processing your Immediate Response query with the Batch Simulator. See the section Batch Simulator.

Workstation Client Platforms (IMS)

VISION:Inform accepts queries and tasks from workstation client platforms to extract subsets of data from host databases. It also accepts control terminal statements from Option 1 (Operations Facilities) and Option 2 (Administration Facilities) on the Main Menu to control system resources and store security profiles in the foreground library.

Figure 1-5 illustrates the interactions of the VISION:Inform components with each other and with IMS.

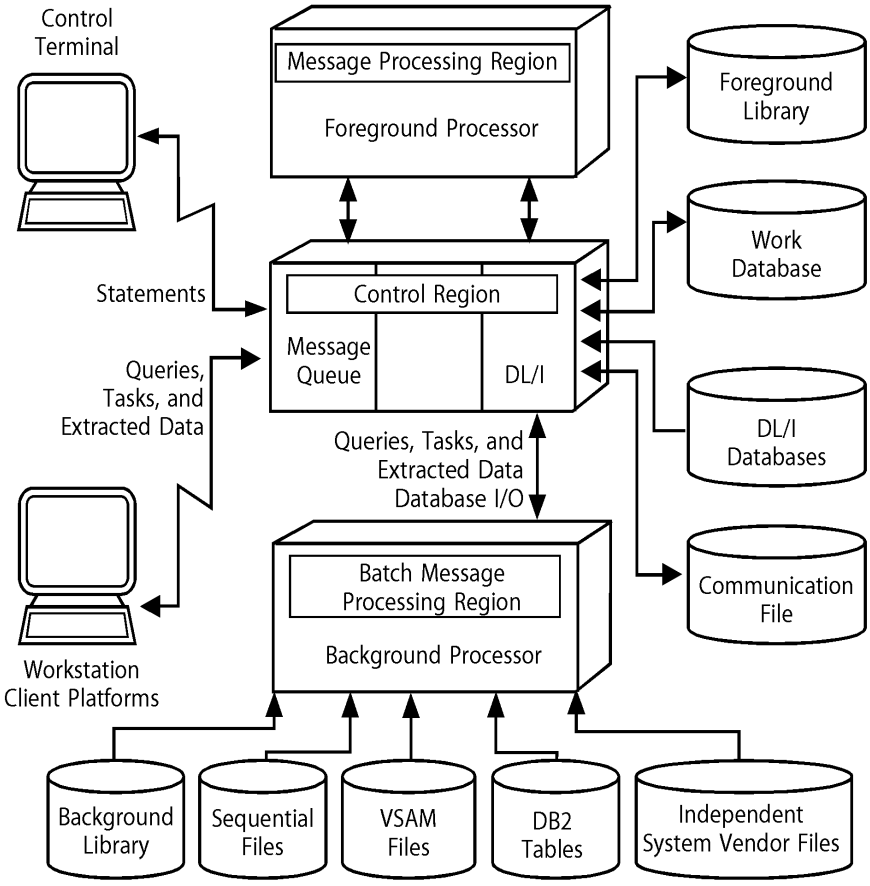


Figure 1-5 Workstation Client Platforms (IMS)

As the system administrator, you enter control terminal statements into the system through a terminal connected to IMS on the host and enter the appropriate logon to establish communications with IMS. After you log on to VISION:Inform, you select an option from the Main Menu in order to enter commands through VISION:Inform.

Note: *See the Main Menu in* Figure 2-2.

VISION:Inform uses DL/I to process control terminal statements that require access to any of its DL/I databases. Any output generated by the statements returns to the terminal through the message queue. Control terminal statements destined for Background Processors are sent by way of the communication file.

When the workstation user orders a query or task, the client program establishes communication with IMS on the host. Once communication is established, the client program sends the appropriate logon sequence, including the VISION:Inform logon.

The logon reaches VISION:Inform by way of the message queue. A valid logon enables the client program to transmit the ordered query or task to the host.

Queries, tasks, and control terminal statements are scheduled by the IMS control region for submission to the Foreground Processor in the MPP region. The region controller activates the Foreground Processor. Queries, tasks, and control terminal statements are sent from the message queue to the Foreground Processor.

Foreground Processor Actions

The Foreground Processor:

- Translates, decodes, and checks the syntax of the queries, tasks, and control statements.
- Uses the foreground library, containing descriptions of the files or databases and the user security profiles, to perform the decoding and syntax checking.
- Issues DL/I calls to access the foreground library and work database. DL/I, residing in the IMS control region, handles these calls.
- Writes the decoded queries and tasks to the communication file, by means of DL/I, where they are held until a Background Processor becomes active.
- When the workstation user requests output generated from a query, the Foreground Processor retrieves it from the communication file and routes it through the message queue to the appropriate user.

Background Processor Actions

- Background Processors are assigned to process various classes of requests by parameters in the INFIN file.
- The appropriate Background Processor reads the decoded query or task from the communication file and batches it, if possible, with other queries using the same database or file.
- Uses the background library to access definitions of fields used during processing.
- When the Background Processor runs in a BMP and issues DL/I calls to access the data stored in the user databases, the foreground library, the work database, and the communication file. DL/I, residing in the control region, handles the calls.
- Stores the output generated from a query in the communication file through DL/I.

CICS System Flow

This section describes how VISION:Inform operates and uses its components in the CICS environment.

The Foreground Processor operates as a CICS transaction. It is activated in the same manner as other CICS transactions.

When you submit a query or task under CICS, the query or task input enters the system through a terminal connected to CICS on the host. The query statements are transferred to the Foreground Processor, which decodes them and interactively checks them for syntax and consistency.

The Foreground Processor uses the foreground library to validate the query or task against file definitions and logical data view definitions. If the Foreground Processor detects a problem, a diagnostic message displays on your terminal or workstation.

During your session, VISION:Inform uses CICS temporary storage to save information between interactions with the control terminal.

VISION:Bridge (CICS)

Note: *See the Main Menu in* Figure 2-2.

With the VISION:Bridge component of VISION:Inform, 3270 terminal users can retrieve and manipulate data, format easy-to-read reports, and create subfiles.

To accomplish this, it uses the Foreground Processor to work with queries built through either Option 4 (Quick Query) or Option 6 (Standard Query Processing) from the Main Menu and one or more Background Processors.

Figure 1-6 illustrates VISION:Bridge and the interaction of its components within CICS.

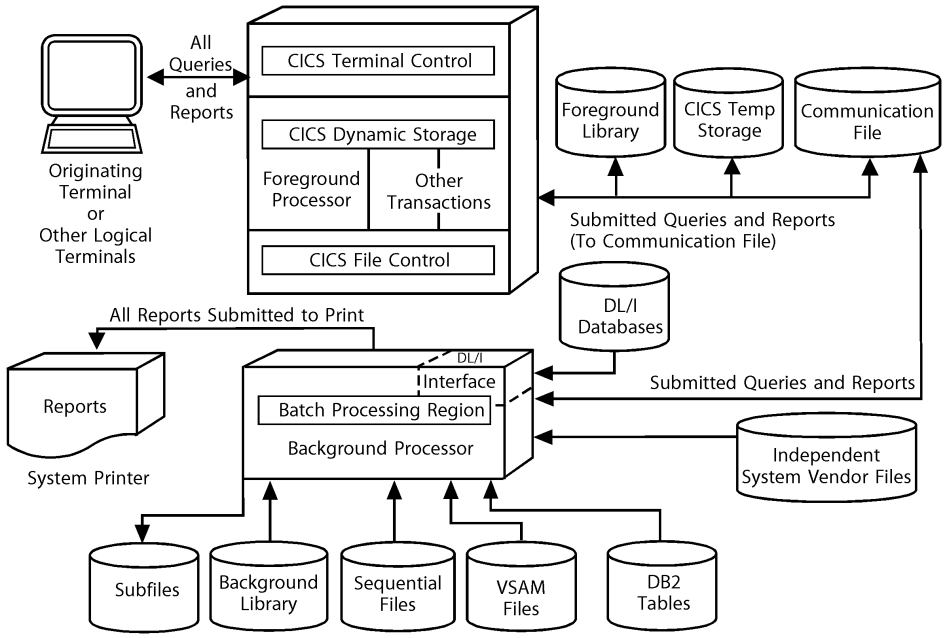


Figure 1-6 VISION:Bridge (CICS)

The communication file is the medium used to transfer queries and reports between the online and batch regions.

A Background Processor, appropriate for the query class, retrieves submitted queries from the communication file and processes them. The Background Processor operates in the batch regions and processes classes 1-14. Figure 1-6 shows one Background Processor, but you can have one or more Background Processors operating in several batch regions.

Foreground Processor Actions

The Foreground Processor:

- Is a pseudo conversational CICS transaction. The CICS transaction passes the queries, tasks, and control statements to it for decoding and syntax checking.
- Translates, decodes, and checks the syntax of the queries, tasks, and control statements.
- Uses the foreground library, containing descriptions of the files or databases and the user security profiles, to perform decoding and syntax checking.
- Uses CICS temporary storage to save information between interactions with the control terminal or workstation.
- Writes the decoded queries and tasks to the communication file where they are held until a Background Processor becomes active.

Background Processor Actions

The Background Processor:

- Translates your query and batches it, if possible, with other queries using the same database or file.
- Uses the background library to access definitions of fields used during processing.
- Retrieves the requested data and manipulates it according to the specifications in the query. If the query requests that the data be sorted, the Background Processor calls the sort program, which sorts the data.
- Stores the output generated by a query in the communication file where you can request to view it at your terminal, route it to other terminals and printers, or purge it.
- Writes subfiles to data sets as they are created.

Background Processor Activation

When a Background Processor for the particular class and data view becomes active, it processes submitted queries.

- Background Processors can become active when there are queries to process.
- Background Processors can also remain active and check the communication file at predefined time intervals for queries awaiting processing.
- Background Processors remain active to process multiple batches of queries until the maximum time has elapsed (as specified in the Background Processor parameters) or the TERM command is issued. The TERM command is only available to the SYSTEM user ID.

Immediate Response (CICS)

Notes:

- *"Immediate Response" was previously referred to as "Inquiry."*
- *See the Main Menu in* Figure 2-2.

The Foreground Processor/Immediate Response Processor retrieves and displays data interactively, without the need for a Background Processor.

You build Immediate Response queries by selecting either Option 5 (Quick Query Immediate Response) or 7 (Immediate Response Query processing) on the Main Menu.

Figure 1-7 illustrates Immediate Response and the interaction of its components within CICS.

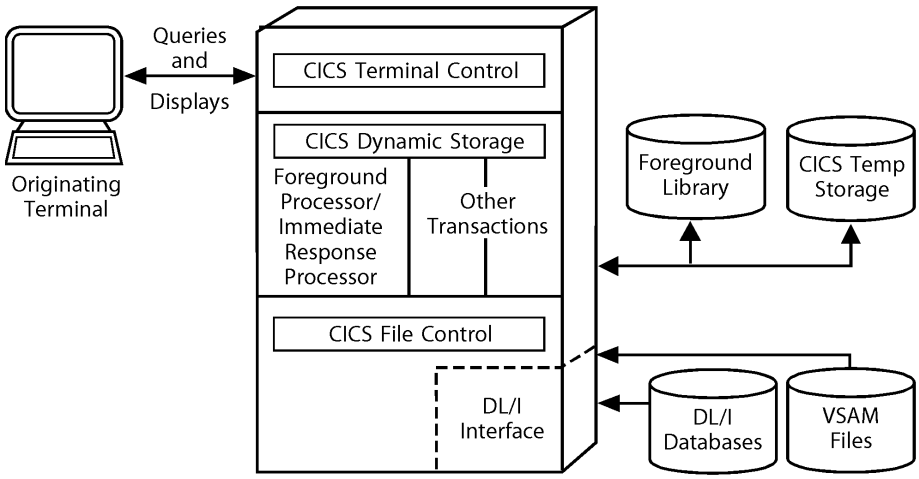


Figure 1-7 Immediate Response (CICS)

Foreground Processor/Immediate Response Processor Actions

The Foreground Processor/Immediate Response Processor:

- Processes complete queries, DISPLAY statements, and Summary statements in the Immediate Response mode.
- Uses the foreground library to access the definitions of fields used in processing.
- Uses the CICS file control program to retrieve the data requested in your query.
- Retrieves data only from DL/I databases (using the CALLDLI interface) and VSAM files.
- Terminates processing of the query, if it encounters problems retrieving data. The query statements display on your terminal with the appropriate diagnostic messages.
- Retrieves the requested data and manipulates it according to the specifications of the query. If the query requests that the data be sorted, the processor calls the sort program, which sorts the data.
- Immediately returns the output generated by complete queries or DISPLAY and Summary statements to the originating terminal.

  In Immediate Response, you cannot route output to other terminals. However, you can obtain hard copy output by running your Immediate Response query with the Batch Simulator. See the section Batch Simulator.

Workstation Client Platforms (CICS)

VISION:Inform accepts queries and tasks from workstation client platforms to extract subsets of data from host databases.

It also accepts control terminal statements, from Options 1 (Operations Facilities) and Option 2 (Administration Facilities) on the Main Menu, to control system resources and to store security profiles in the foreground library.

Figure 1-8 illustrates the interactions of VISION:Inform components with each other and with CICS.

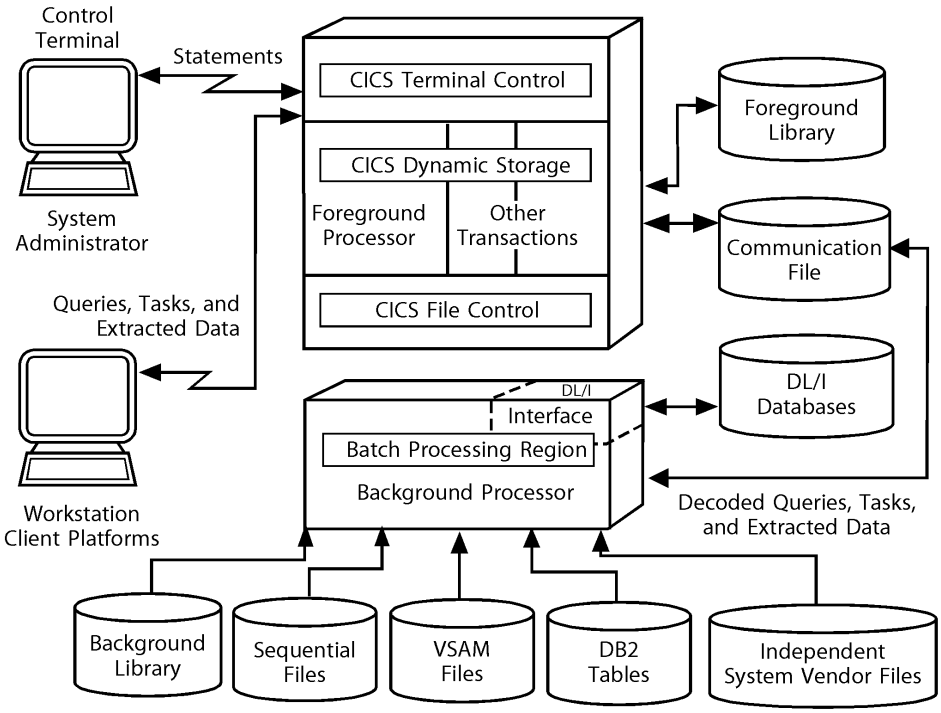


Figure 1-8 Workstation Client Platforms (CICS)

As the system administrator, you enter control terminal statements into the system through a terminal connected to CICS on the host. After you log on to VISION:Inform, you select an option from the Main Menu in order to enter commands through VISION:Inform.

When the workstation user orders a query or task, the client program establishes communication with CICS on the host. Once communication is established, the client program sends the appropriate logon sequence, including the VISION:Inform logon. The logon reaches VISION:Inform by way of CICS terminal control. A valid logon enables the client program to transmit the ordered query or task to the host.

Foreground Processor Actions

The Foreground Processor:

- Is a pseudo conversational CICS transaction. The CICS transaction passes the queries, tasks, and control statements to the Foreground Processor for decoding and syntax checking.
- Translates, decodes, and checks the syntax of the queries, tasks, and control statements.
- Uses the foreground library, containing descriptions of the files or databases and the user security profiles, to perform decoding and syntax checking.
- Uses CICS temporary storage to save information between interactions with the control terminal or workstation.
- Writes the decoded queries and tasks to the communication file where they are held until a Background Processor becomes active.
- When the workstation user requests the extracted data, the Foreground Processor retrieves it from the communication file and sends it to the designated location.

Background Processor Actions

The Background Processor:

- Reads the decoded query or task from the communication file and batches it, if possible, with other queries and tasks using the same database or file.
- Uses the background library to access the file definition information needed to extract the data.
- Accesses the user files or databases. The Background Processor accesses DL/I databases using the DL/I interface and DB2® tables using the DB2 interface.
- Stores the extracted data in the communication file so that it is available to the Foreground Processor.

Batch Simulator

The Batch Simulator provides the same capabilities in the batch environment as are available in the online environment.

The Batch Simulator:

- Submits queries from a batch region and stores them in classes 1–14 in the communication file just as if you had submitted them from an online system. A Background Processor, operating in another batch region, subsequently batches these queries, if possible, and processes them.
- Processes class 15 queries immediately and does not store them on the communication file. There fore, no other background processor is needed.

Using the Batch Simulator

The JCL required for running the Batch Simulator comes with your VISION:Inform system. Insert your logon and query statements in this set of JCL. You compose queries just as you would online, but note the following:

- There are no system prompts to remind you which mode is active. You must keep track of this yourself.
- Each query statement is treated as a new line of the query.
- A blank line is equivalent to a null statement at the terminal. You can insert a blank line at the end of a query to switch to edit mode where you can submit the query.
- The maximum length of a query statement is 72 characters.
- There is no Logon screen, Main Menu screen, or Full Screen Editor in the Batch Simulator.
- The first statement in a Batch Simulator query must be a LOGON command. The format for the command is:

  LOGON USERID userid PASSWORD pswd

  You can omit the keywords USERID and PASSWORD if you enter the userid and pswd in the order shown.

Figure 1-9 shows the system flow of a Batch Simulator query.

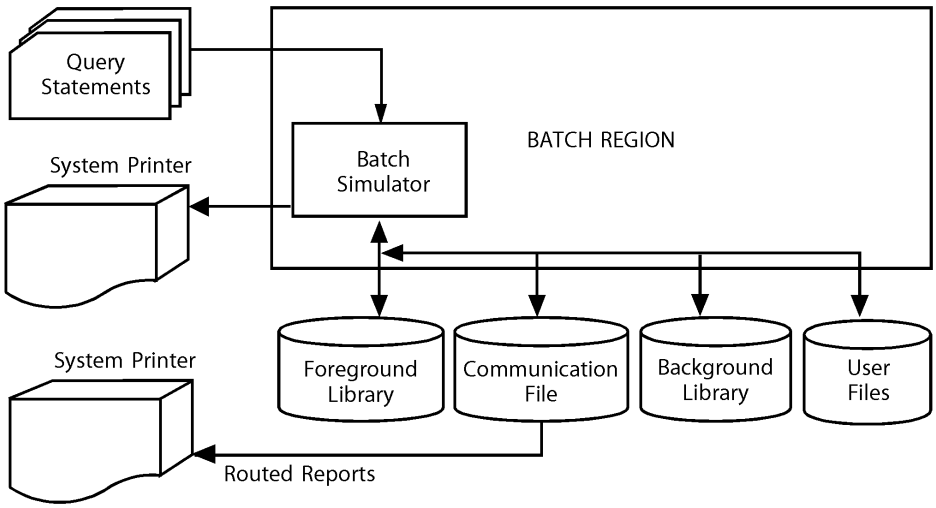


Figure 1-9 VISION:Bridge (Batch Simulator With Non-DL/I)

Figure 1-10 and Figure 1-11 show the source of queries prepared for the Batch Simulator using VISION:Bridge and Immediate Response commands, respectively.

```
LOGON SYSTEM
EDIT TYPE QUERY
QUERY DATABASE CUSTOMER
REPORT CUSTNO CUSTNAME
TITLE 'LIST CUSTOMER FIELDS'
END REPORT
END QUERY

SUBMIT
QUIT
```

Figure 1-10 VISION:Bridge Batch Simulator Query

```
LOGON SYSTEM
IMMED
QUERY DATABASE CUSTOMER
DISPLAY CUSTNO CUSTNAME,
TITLE 'LIST CUSTOMER FIELDS'
QUIT
```

Figure 1-11 Immediate Response Batch Simulator Query

Batch Simulator Actions

The Batch Simulator is activated as an application program in a batch region.

- The Batch Simulator checks the logon command in the INFIN file against the user profile stored in the foreground library.
  - If your logon is valid, the Batch Simulator continues.
  - If your logon is invalid, the Batch Simulator rejects any query statements which follow until it finds a valid logon command.
- When you enter query statements, the Batch Simulator decodes the statements, performs syntax checking, and checks for consistency.
- Next, the Batch Simulator uses the foreground library to validate the query against file definitions and logical data view definitions.

  If a problem is detected, the Batch Simulator rejects the query and issues the appropriate diagnostic message. The Batch Simulator then processes the next statement in the input stream.
- After the Batch Simulator processes the statements, the Batch Simulator submits the query to the communication file for processing by a Background Processor.
- The Batch Simulator terminates when there are no more commands in the input stream. Multiple sessions can be present in a single input stream of the Batch Simulator.

Background Processor Activation

When a Background Processor for the particular class and data view becomes active, it processes the submitted queries.

- Background Processors can become active when there are queries to process.
- Background Processors can also remain active and check the communication file at predefined intervals for queries awaiting processing.
- Background Processors can be executed when there are queries to process or they can execute, remain active, and check the communication file at predefined time intervals for queries awaiting processing.

For detailed information on using the Batch Simulator, see Chapter 6, "Using the Batch Simulator".

Contacting Technical Support

For further 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.

Chapter

2

Using VISION:Inform

This chapter describes accessing and using VISION:Inform. The VISION:Inform user interface tools are:

- Logon Panel.
- Main Menu.
- Source Processing Panel.
- Full Screen Editor Panel.
- Line Commands.
- Primary Commands.
- Help Panels.
- Diagnostic Messages.

Accessing VISION:Inform

Access to VISION:Inform depends upon the teleprocessing system that the product runs under.

- For CICS systems, connect and log on to your CICS online system in the usual manner. Then, from a clear screen, enter the transaction that was associated with the program INFORMOL during system installation. The default transaction code for this is INFP.
- For IMS systems, connect and log on to your IMS online system in the usual manner. Then, from a clear screen, enter the /FORMAT IMS command, using the format associated with the initial output Logon panel in the MFS Logon panel source. The default format specification for 3270 Model 2 connection to VISION:Inform is /FORMAT INX202. Other default formats are listed in the *VISION:Inform Installation Guide* for your environment.

After performing the access described above for your environment, the Logon panel appears so you can log on with your VISION:Inform user ID.

Logon Panel

Log on to VISION:Inform by entering your user ID and password on the Logon panel (shown in Figure 2-1) and pressing Enter. The system administrator's ID, SYSTEM, is referred to as SYSTEM user ID.

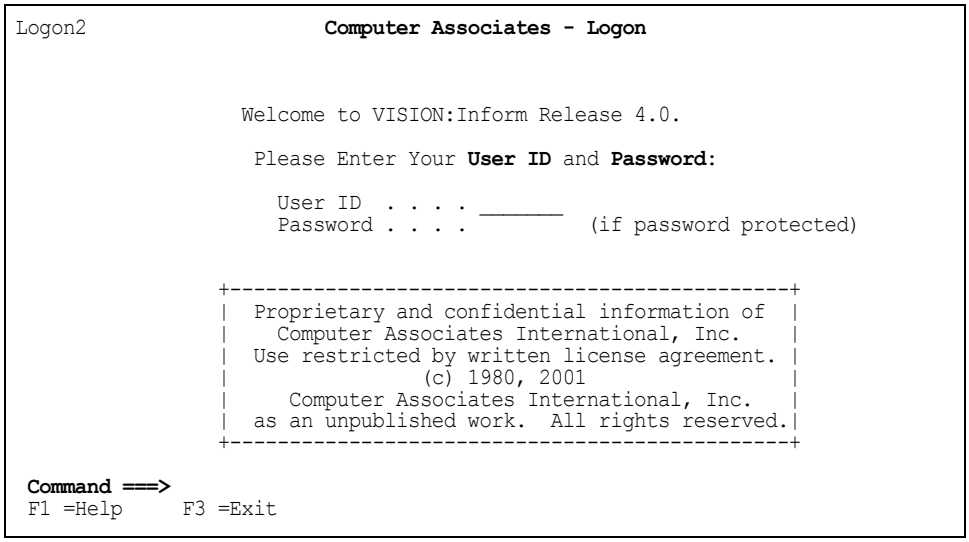


Figure 2-1 Logon Panel

Main Menu

After entering your user ID, the VISION:Inform/VISION:Bridge Main Menu appears (shown in Figure 2-2).

```
Menu2                Computer Associates - Main Menu


_ Enter one of the following VISION:Inform or VISION:Bridge Options:

  VISION:Inform Options:

  1. Operation Facilities                    (Background Processor Status)
  2. Administration Facilities               (Profile Development)
  3. Report Facilities                       (Report Handling)

  VISION:Bridge Options:

  4. Quick Query                             (Assisted Query Development)
  5. Quick Query Immediate Response          (Assisted Query Development)
  6. Standard Query Processing               (Submit, Delete, Edit
                                               Queries and Stmts)
  7. Immediate Response Query Processing     (Run Queries and Immed Mode)


Command ===>
F1 =Help     F12=Cancel
```

Figure 2-2 Main Menu

Use the Main Menu to select options:

- Use Options 4 through 7 for VISION:Bridge.
- Use Options 1 through 3 for VISION:Inform system administration activities.
- Use Options 1, 3, and 7 to display a Command Input screen where you can enter commands to examine and control the execution of queries, tasks, and Background Processors.

  A Command Input screen is designated by a ?: prompt in the upper left corner of the screen. You can enter the commands through the options described in Chapter 2, "Using VISION:Inform". Enter the QUIT command to exit from a Command Input screen.
- Use Option 2 (Administration Facilities) for creating user profiles. This option is only accessible by the SYSTEM user ID. User profiles are discussed in Chapter 3, "VISION:Inform System Security". The panels for creating profiles are discussed in the following sections.

Source Processing Panel

To create or edit a user profile, select Option 2 (Administration Facilities) on the Main Menu to display the Source Processing panel (shown in Figure 2-3).

```
Source2 --------------- Computer Associates - Source Processing ------------
                                                                More:   - +
   Name     Type    Owner   Last Used      Name     Type    Owner   Last Used
_ SYSTEM   PROFILE          06/23/01    _ S17      PROFILE  SYSTEM   06/23/01
_ SYSXTRA  PROFILE  SYSTEM   11/18/99    _ S18      PROFILE  SYSTEM   08/10/00
_ SYS1     PROFILE  SYSTEM   06/24/00    _ S19      PROFILE  SYSTEM   06/23/01
_ SYS1LOGO PROFILE  SYSTEM   09/04/99    _ S2       PROFILE  SYSTEM   06/23/01
_ SYS2     PROFILE  SYSTEM   04/14/00    _ S20      PROFILE  SYSTEM   08/10/00
_ SYS3     PROFILE  SYSTEM   04/20/00    _ S21      PROFILE  SYSTEM   08/10/00
_ S0       PROFILE  SYSTEM   08/10/00    _ S22      PROFILE  SYSTEM   08/10/00
_ S0000100 PROFILE  SYSTEM   04/05/00    _ S24      PROFILE  SYSTEM   06/23/01
_ S0105663 PROFILE  SYS1     10/01/99    _ S25      PROFILE  SYSTEM   08/10/00
_ S1       PROFILE  SYSTEM   06/23/01    _ S26      PROFILE  SYSTEM   08/10/00
_ S10      PROFILE  SYSTEM   06/23/01    _ S27      PROFILE  SYSTEM   06/23/01
_ S11      PROFILE  SYSTEM   06/23/01    _ S28      PROFILE  SYSTEM   06/23/01
_ S12      PROFILE  SYSTEM   06/23/01    _ S29      PROFILE  SYSTEM   06/23/01
_ S13      PROFILE  SYSTEM   06/23/01    _ S3       PROFILE  SYSTEM   08/10/00
_ S14      PROFILE  SYSTEM   08/10/00    _ S30      PROFILE  SYSTEM   08/10/00
_ S15      PROFILE  SYSTEM   06/23/01    _ S31      PROFILE  SYSTEM   08/10/00
_ S16      PROFILE  SYSTEM   06/23/01    _ S32      PROFILE  SYSTEM   06/23/01

Command ===>
F1 =Help     F7 =Backward F8 =Forward  F12=Cancel
```

Figure 2-3 Source Processing Panel

The Source Processing panel displays the name of each profile in the foreground library. From this panel, you can create and edit user profiles, or delete user profiles from the foreground library. If you have just installed your VISION:Inform, only the SYSTEM profile appears.

Each profile is preceded by an underscore; this is where you specify what you want to do with the profile. The following codes are valid:

E To edit the profile. This displays the Full Screen Editor panel containing the specified profile.

D To delete the profile from the foreground library. After entering the D and pressing Enter, a prompt asks you to re-enter the D to confirm the delete. This helps prevent accidental deletion.

Code only one entry (E or D) per interaction. Use the LOCATE primary command to position the cursor in the Source Processing panel to a specific profile.

Notes:

- *You can abbreviate Command line entries to the shortest string which uniquely identifies the command or keyword.*
- *For example, CREATE name PROFILE can be abbreviated to CRE name PR.*
- *Items are profile, queries, or statements.*

To create a new profile, move the cursor to the Command line and enter:

```
CREATE name PROFILE
```

where:

CREATE Specify the command.

name Specify the profile name, which becomes the user ID required to gain access to the system.

PROFILE Specify the type of item you are creating.

The CREATE command displays the Full Screen Editor panel, as does editing (E) an existing profile.

Full Screen Editor Panel

Using the Full Screen Editor panel to create or change profiles, you can:

- Scroll the data up, down, left, or right.
- Change the data by typing over the data being displayed and by using line commands and primary commands.
- Enter line commands directly over the line number of the lines to be affected (see Line Commands).
- Enter primary commands on the Command line (see Primary Commands).

To display the Full Screen Editor panel, select a profile to edit or enter the CREATE profile primary command in the Source Processing panel.

When you create a new profile, the Full Screen Editor panel is initially empty.

Full Screen Editor Panel Components

The Line Command area on the left side of the panel contains a series of lines (indicated by dots . . . . .) for a new profile. When you enter information on the lines, sequence numbers replace these dots. You specify line commands in the Line Command area.

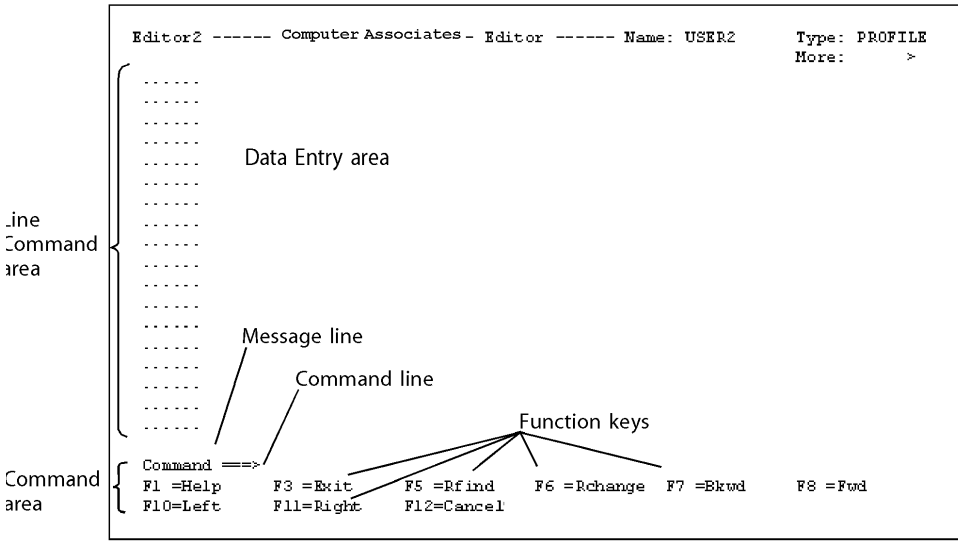


Figure 2-4 Creating a New Profile

The area to the right of the Line Command area is the data entry area where you enter the statements for the profile being edited.

The message line displays informational and diagnostic messages during your edit session.

The last three lines of the panel are the Command area. The Command area includes the Command line and two function key lines.

- You enter primary commands on the Command line.
- The function key lines contain the function keys that are available for use on the panel. The section titled Primary Commands identifies function keys associated with selected primary commands.

Copy a Profile

To copy an existing profile to use as a template for a new profile, move the cursor to the Command line and enter:

COPY name PROFILE

This command places a copy of the existing profile into the data entry area where you can modify it to make a new profile.

When you are editing an existing profile, the Full Screen Editor panel displays the source statements of the item with sequence numbers.

The following Full Screen Editor panel shows the SYSTEM profile as it is delivered with VISION:Inform.

```
Editor2 ----- Computer Associates - Editor ------ Name: SYSTEM   Type: PROFILE
                                                                  More:      >
000100 PROFILE
000200 END PROFILE









Command ===>
F1 =Help      F3 =Exit     F5 =Rfind    F6 =Rchange  F7 =Bkwd     F8 =Fwd
F10=Left      F11=Right    F12=Cancel
```

Figure 2-5 Full Screen Editor Panel

The first line of the panel contains the name of the profile along with the item type (PROFILE). The second line contains the scrolling information field “More:”, which indicates the presence of the following data:

< Left of the data entry area shown.

> Right of the data entry area shown.

- Above the data entry area shown (backward).

+ Below the data entry area shown (forward).

Note that blanks are considered data, and all edited items appear as fixed length 132 character records.

Scrolling

To view the area that is not currently shown, enter the appropriate scrolling command. The scrolling commands are: LEFT, RIGHT, BACKWARD, and FORWARD. You can also use these commands with optional parameters. For instance:

FORWARD n (where n is a number from 1 – 999999)

Use the FORWARD command parameter to specify how many lines forward to scroll the window.

LEFT M (M stands for maximum)

Use the LEFT command parameter, M, to specify scrolling the window as far left as possible.

Scrolling commands are primary commands and can be assigned to function keys. For a list of primary commands and applicable function keys, see the section Primary Commands.

Commands and function key assignments are specified in the PARMBLK CUSTOM macro as described in VISION:Inform *Installation Guide* for your environment.

Line Commands

Note: *This section refers to the default values delivered with the system.*

Line commands affect only a single line or a block of lines. You enter line commands by typing them in the Line Command area on one or more lines and pressing Enter.

The following lists the line commands and a description of each.

Line Command	Description
A	Identifies the line after which moved or copied lines are to be inserted. Use in conjunction with the M line command, the C line command, or the COPY primary command.
B	Identifies the line before which moved or copied lines are to be inserted. Use in conjunction with the M line command, the C line command, or the COPY primary command.
C	Copies this line to the location specified by the A or B line command.
Cn	Copies n (where n is a number from 1 – 99999) lines to the location specified by the A or B line command.
CC	Delimits a block of lines to be copied to the location specified by the A or B line command.
D	Deletes this line.

Figure 2-6 Line Commands (Page 1 of 2)

Line Command	Description
Dn	Deletes n (where n is a number from 1 – 99999) lines.
DD	Delimits a block of lines to be deleted.
I	Inserts a blank line after this line.
In	Inserts n (where n is a number from 1 – 99999) blank lines after this line.
M	Moves this line to the location specified by the A or B line command.
Mn	Moves n (where n is a number from 1 – 99999) lines to the location specified by the A or B line command.
MM	Delimits a block of lines to be moved to the location specified by the A or B line command.
R	Repeats this line on the next line.
Rn	Repeats this line n (where n is a number from 1 – 99999) times immediately following the line on which Rn is entered.
RR	Delimits the block of lines to be repeated immediately following the second RR command.
RRn	Delimits the block of lines to be repeated n (where n is a number from 1 – 9999) times immediately following the second RRn command. (You can enter the n on either or both of the RR commands.)

Figure 2-6 Line Commands (Page 2 of 2)

Primary Commands

Primary commands affect the entire item (profile, query, or statements) being edited. Enter Primary commands in one of the following ways:

- Enter the command in the Command area (Command ===>) and press Enter. You can abbreviate primary commands to the shortest string which uniquely identifies the command.
- Press a function key that has been defined to execute a specific command.

Notes:

- *You can use primary commands when editing profiles (Option 2 Administration Facilities) or queries and STMTS (Option 6 Standard Query Processing).*
- *Command syntax often depends upon the type of item you are editing.*

The following sections describe the primary commands available in the VISION:Inform Full Screen Editor and a description of each. The default function key is also shown, if applicable.

BACKWARD Command (F7)

Scrolls the window toward the top of the data. You can include the optional parameters of M (maximum) or n (where n is a number from 1 – 999999).

CANCEL Command (F12)

Ends the current edit session without saving the item, and returns to the calling panel.

CHANGE Command

Changes the first or ALL occurrences of one text string to a second text string. The format of the command is:

CHANGE string1 string2 (ALL)

If you specify ALL, all occurrences of the specified string are changed.

If you do not specify ALL, only the first occurrence changes.

If the cursor is in the data entry area, the search begins at the cursor position.

If the cursor is not in the data entry area, the search starts at the beginning of the data entry area.

Use single (') or double (") quotation marks as delimiters to surround string1 and string2. Quotation marks are only necessary when these delimiting characters are used in the data or blanks are part of either string.

- Changed lines are marked by ==>CHG in the Line Command area. The lengths of string1 and string2 can be unequal.
- The lengths of string1 and string2 are limited by the length of the Command line area for your terminal. String1 cannot be null.

COPY Command

Copies a profile, query, or collection of statements from the foreground library to the location specified by the A or B line command. (The A or B line command is not required if the item being edited is empty.) The format of the command is:

```
COPY name type
```

where name is the name of the profile you want to copy, and type is either PROFILE, QUERY, or STMTS.

CREATE Command

Saves the item in the current edit session under a different name or type as specified in the command. The format of the command is:

```
CREATE name type
```

where name is the name of the item you want to create, and type is PROFILE, QUERY, or STMTS.

The item being saved is validated for syntax and internal consistency unless the type is STMTS.

DATAVIEW Command

Displays a panel containing a list of available data views for this VISION:Inform user ID. The DATAVIEW command is not available when editing profiles.

EXIT Command (F3)

Saves the current item and returns to the calling panel. If the item does not successfully validate, a message appears indicating that you must make corrections or use the CANCEL command to exit the session.

FIELDS Command

Displays a panel containing fields of the data view of the query currently being edited.

You must validate the query at least once before this command is available. (See the sections SAVE Command and the VALIDATE Command.)

When you select a field from the Fields panel, detailed information for that field displays.

The FIELDS command is not available when editing profiles.

FIND Command

Finds the first occurrence of a text string in the data entry area. The format of the command is:

```
FIND string
```

If the cursor is in the data entry area, the search begins at the cursor position.

If the cursor is not in the data entry area, the search starts at the beginning of the data entry area.

Use single (‘) or double (“) quotation marks as delimiters to surround the string. They are only necessary when these delimiting characters are used in the string or blanks are part of the string.

Note: *The length of the string is limited by the length of the Command line area for your terminal.*

FORWARD Command (F8)

Scrolls the window toward the bottom of the data. You can include the optional parameters of M (maximum) or n (where n is a number from 1 – 999999).

HELP Command (F1)

Displays a Help panel based on the location of the cursor. (For example, if the cursor is on the Command line, the Help panel contains Command line Help information.)

LEFT Command (F10)

Scrolls the window toward the left margin. You can include the optional parameters of M (maximum) or n (where n is a number from 1 – 999999).

LOCATE Command

Note: *The LOCATE command has a slightly different format (LOCATE string) in other panels to position a list to a particular location. For additional information, see the* VISION:Inform *User Guide.*

Locates a specific line of data based on the parameter specified. Following are samples of the command format:

LOCATE n (n is 1 – 999999)
LOCATE ERR
LOCATE CHG

If you specify n and an exact match of that line number cannot be found, the row with the line number closest to, but less than that number will be the new first row in the data entry area.

If you specify ERR, the cursor is positioned on the next line with the ==>ERR (validation error) flag.

If you specify CHG, the cursor is positioned on the next line with the ==>CHG (changed data) flag.

If you specify ERR or CHG, the search begins on line two of the data entry area.

RCHANGE Command (F6)

Repeats the last CHANGE command that was entered. (See the CHANGE Command.)

RENUMBER Command

Renumbers all lines. The first line of the data is assigned a sequence number of 100 and the remaining lines are incremented by 100.

RESET Command

Clears any pending line commands, messages in the message line, statement diagnostic messages, and ==>CHG indicators.

RFIND Command (F5)

Searches for the string entered in the last FIND or CHANGE command. (See the FIND and CHANGE commands.)

RIGHT Command (F11)

Scrolls the window toward the right margin. You can include the optional parameters of M (maximum) or n (where n is a number from 1 – 999999).

SAVE Command

Saves the item without terminating the edit session. The item is saved in the foreground library under the name and type as shown in the first line of the panel.

The item is validated prior to the save unless the parameter ASIS is specified.

SUBMIT Command

Available only when editing queries. The SUBMIT command displays the Submit panel, so the query being edited can be written to the communication file for execution by a Background Processor.

VALIDATE Command

Validates the query or profile in the current edit session. (STMTS are not validated.) If syntax or consistency problems are found, a message displays following the lines in question and the line is flagged with ==>ERR in the line command area.

Help Panels

Online Help panels give you specific information regarding the panel you are currently using. To request Help, press F1 (the default key assignment) or enter HELP in the Command area.

The information in the Help panel is based on the current cursor location. That is, if the cursor is in the Line Command area of the Full Screen Editor panel, the Help panel contains Line Command information. Or, if the cursor is on the Main Menu, information is displayed regarding the Main Menu.

Help panels are available from all VISION:Inform panels, except the Command Input panel and the Help panels themselves.

The following shows the Help panel displayed from the Main Menu.

```
 Help2          Computer Associates - Help: Main Menu (body area)


A valid option (1-7) must be entered in the single field provided on this
screen. Valid VISION:Product options are as follows:

1 - Operation Facilities - For status updates on background processors, and
    for determining the System Maintenance Level.
2 - Administration Facilities - For creating, updating, and deleting User
    Profiles.
3 - Report Facilities - For Query status, and Query output processing.

Valid VISION:Bridge options are as follows:

4 - Quick Query - For assisted development of VISION:Bridge queries.
5 - Quick Query Immediate Response - For assisted development of
    interactive queries.
6 - Standard Query Processing - For submitting queries, and for creating,
    updating, and deleting items of type QUERY and STMTS.
7 - Immediate Response Query Processing - For interactive query processing.

Command ===>
F12=Cancel
```

Figure 2-7 Help Panel

To exit from any Help panel and return to the previous panel, enter CANCEL on the Command line or press the function key assigned to the CANCEL command. The default is F12 for CICS or F24 for IMS.

Diagnostic Messages

The Full Screen Editor validates items being edited for syntax and consistency. When the Full Screen Editor finds a problem, it issues a message.

Validation occurs only at specific points during an editor session.

- For both profile and query editing, use the SAVE, CREATE, EXIT, or VALIDATE primary commands to validate the item.
- For queries, validation also occurs when you use the SUBMIT command.

```
Editor2 ------ Computer Associates - Editor ---- Name: USER2     Type: PROFILE
                                                                 More:      >
000100 PROFILE
000200 INCLUDE CUSTOMER CLASS 17 PRIVATE
000300 END INCLUDE
000400 INCLUDE ITEM
000500 END INCLUDE
000600 INCLUDE CUSTITEM
000700 END INCLUDE
000800 END PROFILE









Command ===> save
F1 =Help      F2 =Reset    F3 =Exit    F4 =Save   F5 =Rfind   F6 =Rchange
F7 =Backward  F8 =Forward  F9 =Submit  F10=Left   F11=Right   F12=Cancel
```

Figure 2-8 Saving a Profile with Problems

As an example, assume you have entered the profile in and then entered the SAVE command. The resulting panel is shown in Figure 2-9.

```
Editor2 ------ Computer Associates - Editor ----- Name: USER2    Type: PROFILE
                                                                 More:       >
==>ERR INCLUDE CUSTOMER CLASS 17 PRIVATE
IU02** CLASS VALUE SPECIFIED NOT WITHIN THE ALLOWABLE RANGE OF       1 TO 14
==>ERR END INCLUDE
PE03** THE END STATEMENT IS OUT OF CONTEXT.
000400 INCLUDE ITEM
000500 END INCLUDE
000600 INCLUDE CUSTITEM
000700 END INCLUDE
000800 END PROFILE




IY02** Item USER2     type PROFILE  contains errors and was not saved.
Command ===>
F1 =Help     F2 =Reset    F3 =Exit    F4 =Save    F5 =Rfind    F6 =Rchange
F7 =Backward F8 =Forward  F9 =Submit  F10=Left    F11=Right    F12=Cancel
```

Figure 2-9 Diagnostic Messages

The Full Screen Editor responds by moving the first line that did not validate to the top of the panel and placing an indicator (==>ERR) in the line command of each such line. Directly beneath each such line is a message or messages describing what is wrong.

In this example, the CLASS value is invalid. This causes the entire INCLUDE statement to be invalid, which, in turn, causes the END INCLUDE statement to be out of context.

The message at the bottom of the panel indicates that the profile could not be saved, because it contains errors.

Chapter

3

VISION:Inform System Security

By means of the user profile, VISION:Inform provides you with the ability to protect databases from unauthorized access at five different levels:

- At the operating system level, VISION:Inform can interface with your MVS, OS/390, or z/OS security system (CA-ACF2, CA-Top Secret, RACF, and so on) for password validation, using the VISION:Inform Profile Exit routine..
- At the user level, the system administrator can specify user access to the system through the use of private user IDs and passwords. The user must enter the correct ID and password to use the system.
- At the database level, the system administrator can give or deny users access to entire databases.
- At the segment and field level, the system administrator can restrict access to certain segments and fields of a database.
- At the record level, the system administrator can restrict access depending on the values of specific fields.

These security features are implemented through user profiles and the Profile Exit facility. Profiles act as internal lists of characteristics which determine what data and processing options are available to each user.

Profiles can only be created and maintained by a special system administration ID called the SYSTEM user ID. This ID is delivered with VISION:Inform.

Since security requirements vary considerably among installations, the VISION:Inform profile capabilities have been designed so that you can tailor your profile hierarchy to the needs of your installation.

User Profiles

User profiles can contain the following information:

- Accessible databases.
- Restricted database segments.
- Restricted database fields.
- Logical expressions defining record selection criteria.
- Passwords for entering the system.
- System processing options.

The information contained in the profile pertains to a specific user ID although multiple users can log on under the same user ID. When a user logs on to VISION:Inform, the previously defined profile for that user is located and the passwords from the logon and the profile are compared. If they do not match or a profile does not exist for that user ID, the logon is rejected.

You, the system administrator, create profiles using the SYSTEM user ID. You enter the profile statements through Option 2 (Administration Facilities) on the Main Menu.

The SYSTEM Profile

The SYSTEM user ID is the system administrator's user ID. It is your only means of logging on to a new VISION:Inform installation. It is also the only user ID authorized to create and change user profiles. SYSTEM user ID has other extended capabilities such as being able to access queries, tasks, and reports belonging to any user of the system.

When you install VISION:Inform, the profile for the SYSTEM user ID is empty, except for the required entries PROFILE and END PROFILE. The SYSTEM profile is the first profile you define. Include a password, as well as the databases and processing options that will be available to the SYSTEM user ID.

Profile Structures

Note: *For additional information on the PROFILE command GROUP parameter, see the section* *PROFILE Command*.

Conceptually and functionally, user profiles exist in a hierarchy. The SYSTEM profile is always at the top and various levels of subordinate profiles lie beneath. Selected characteristics of one particular profile can be passed, if needed, to subordinate profiles through the use of the PROFILE command GROUP

parameter. This gives rise to the concept of PROFILE groups where any number of subordinate profiles inherit characteristics from a higher level profile. Each subordinate profile is able to possess unique characteristics.

PROFILE Group Concepts

The PROFILE group is a powerful feature of VISION:Inform. When processing a query, VISION:Inform first looks at the user's profile for information pertaining to data access and processing options.

Note: *For additional information on PROFILE groups, see the section PROFILE Groups.*

- If the user's profile contains a GROUP parameter followed by the name of another user profile, VISION:Product looks in the specified PROFILE group for additional information about data access and processing options.
- If the PROFILE group has its own GROUP parameter, VISION:Inform applies this set of profile specifications on the original user ID. In this way, you can establish a hierarchy that leads right up to the SYSTEM profile.

Deleting a Higher Level Profile

When you delete a higher level profile, make appropriate changes for consistency. If you do not update a subordinate profile when you delete a higher level profile, VISION:Inform issues a diagnostic message when the subordinate profile user tries to log on.

Hierarchy Levels

There can be any number of levels in the hierarchy.

- Any profile, except the SYSTEM profile, can be both a group profile and a subordinate profile at the same time.
- A profile can inherit characteristics from a higher level profile and simultaneously pass characteristics to subordinate user profiles.
- Parameters on the PROFILE statement itself are not inherited by subordinate profiles.
- The SYSTEM profile cannot be a subordinate profile.

Inheritance Versus Restriction

Conceptually, there are two ways of structuring profile hierarchies: by inheritance and by restriction.

Inherited Structure

In an inherited structure, lower level profiles inherit characteristics from higher level group profiles, including the SYSTEM profile (by default). This results in profiles at the lowest levels in the hierarchy to possess the greatest access to information in the system.

Restricted structure

Restricted structures model the traditional organizational hierarchy. Power is concentrated at the top of the hierarchy and becomes increasingly limited as you move to lower levels.

- The higher the profile in the hierarchy, the fewer the restrictions on that user's access to data in the system.
- Conversely, the lowest level subordinate profiles in a restricted structure have the least access to data.

Some applications are best served by some form of restricted structure, although inherited structures have their uses. You can mix both types of profile structures, but we do not recommend it. In large profile hierarchies, it becomes increasingly difficult to keep track of the individual user profiles and the characteristics they possess.

Inherited Profile Structures

In an inherited profile structure, characteristics defined in a PROFILE group apply to all the profiles below it.

- Profiles at the bottom of the hierarchy inherit characteristics and access to more data than profiles at the top of the hierarchy.
- However, subordinate profiles can still override characteristics which would otherwise be inherited from a PROFILE group and the SYSTEM profile.

The major advantage of an inherited profile structure is the ease with which you can define and update additional profiles.

- For example, if you define access to a database in a PROFILE group, all its subordinate profiles inherit access to that database.
- If you delete access to that database from the PROFILE group, all the subordinate profiles lose access also.

Restricted Profile Structures

Restricted structures are similar to inherited structures. They consist of explicit hierarchies where subordinate profiles can inherit access and processing characteristics from PROFILE groups. The key difference is introduced by the EXCLUDE command and the PRIVATE parameter.

- The INCLUDE command PRIVATE parameter is placed in the PROFILE group and prevents subordinate profiles from inheriting database access from a group profile.
- The EXCLUDE command is placed in the subordinate profile. This denies the user and subordinate profiles access to databases which would otherwise be inherited.

Together, these commands provide the user profile situated at the top of the hierarchy with the most global access to data in the system, as well as the most processing options available.

At each level in the hierarchy (including the SYSTEM profile):

- Use the PRIVATE parameter to selectively prevent subordinate profiles from inheriting selected access and processing capabilities.
- Use the EXCLUDE command to prevent a user from accessing a database to which access would otherwise be inherited from a group profile.

Restricted Structure Example

In a restricted structure, you could have a manager at the top of the hierarchy. This manager has access to every database of relevance for a number of projects.

Under this profile, you could add a level of project leaders, each of whom have access to different subsets of the data available to the manager.

Finally, under each project leader, you could have a level of workers who can have access to a restricted set of each project leader's capabilities. Thus, as you move down the hierarchy, each subsequent level of profiles has fewer options than those above it.

Using the PRIVATE Parameter

Since only the inheritance of access is being prevented by the PRIVATE parameter, you must explicitly define the characteristics for each individual user. This is unlike the inherited structure. If a new database comes online under a restricted hierarchy (that is, in a PRIVATE parameter), you explicitly include the new database in the profile of each user who will access it.

You can build hybrid profile hierarchies in which selected characteristics are passed to subordinate profiles while others are not.

Defining User Profiles

Define the SYSTEM profile first. Using the instructions in Chapter 2, "Using VISION:Inform", log on to VISION:Inform with the SYSTEM user ID. Chapter 2 also contains instructions for creating and updating profiles. When you finish modifying the SYSTEM profile, press F3 to save the profile and exit the panel.

Profile Name

When you complete a profile and save it in the foreground library, the name you assign to the profile becomes the user ID required to gain access to the system. Profile names cannot contain the system delimiter character (default is #) or imbedded blanks.

Verifying the SYSTEM user ID Profile

When you define or change the SYSTEM profile, end your session and log back on to verify your new profile. Remember, only the SYSTEM user ID can create and modify profiles.

Comments in Profiles

You can add comments to all PROFILE statements, except for END statements. Place a semi-colon (;) between the statement and the comment. You can also use comment statements that contain a semi-colon in column 1 in profiles.

Verifying a User Profile

After you save the profiles in the foreground library, verify each profile by logging on to VISION:Inform using each user ID to make sure there are no problems. If you cannot log on with a user ID, it could be in a recursive group arrangement. Log back on with the SYSTEM user ID and check for profiles with groups that are subordinate to other groups. If you find a cyclical arrangement (for example, group A is subordinate to group B, which is subordinate to group A), the circle must be broken before the user IDs involved can be used.

Profile Commands

Notes:

- *Statement = command + parameters.*
- *Parameters are composed of keywords followed by operands.*
- *In selected text, identifiers, specifications, and options can be used interchangeably.*

There are a number of commands available for creating user profiles.

- Each command can have optional and required parameters which combine to form command statements.
- Enter commands and their parameters as statements on one line unless continued with a comma.

There are seven basic profile commands (PROFILE, EXCLUDE, INCLUDE, RESTRICT, SELECT, END INCLUDE, and END PROFILE). There is one additional command (PROHIBIT) for VISION:Bridge and Immediate Response.

This section briefly introduces the commands for creating and maintaining profiles. Following this overview is a section describing each command in detail.

PROFILE Command

The PROFILE command itself provides global information that applies to every task or query submitted by the user, such as group identification and batching information.

PROFILE Groups

All the statements forming a complete user profile combine to form a PROFILE group. PROFILE groups begin with a PROFILE command and end with an END PROFILE command. PROFILE and END PROFILE must be the first and last commands in a user profile.

INCLUDE Groups

Within a PROFILE group you can have one or more INCLUDE groups. Each begins with an INCLUDE command and ends with an END INCLUDE command. Each INCLUDE group gives VISION:Product information about one database that can be accessed by this user ID.

For additional information, see the section PROFILE Command.

INCLUDE Command

The INCLUDE command provides explicit access to databases. It also provides VISION:Product with information about the processing classes available to the user. You can have as many INCLUDE groups in a profile as you need.

INCLUDE has two subcommands: RESTRICT and SELECT.

- RESTRICT limits access to specific fields or segments within a particular database.
- SELECT provides selection criteria which limit access to data at the record level.

Each INCLUDE command in a profile, along with its associated RESTRICT and SELECT commands, forms an INCLUDE group which ends with an END INCLUDE command.

For additional information, see the section INCLUDE Command.

EXCLUDE Command

The EXCLUDE command prevents a user from accessing a database to which access would otherwise be inherited from a group profile.

For additional information, see the section EXCLUDE Command.

Note: *The PROHIBIT command applies to query language statements and command input statements only. You cannot use the PROHIBIT command to prohibit the use of primary commands in panels.*

PROHIBIT Command

The PROHIBIT command prohibits the use of either an entire command or certain keywords of a command. Use the PROHIBIT command only with VISION:Bridge or Immediate Response.

For additional information, see the section PROHIBIT Command.

Profile Commands Relation Summary

The various profile commands relate to one another as summarized below.

```
PROFILE
   EXCLUDE (database)
   INCLUDE (database)
      RESTRICT (segments, fields)
      SELECT (records)
   END INCLUDE
  INCLUDE (database)
  END INCLUDE
END PROFILE
```

Commands and Parameters

The following sections describe the various commands and their parameters. You can use parameters either as a keyword parameter or a positional parameter.

Keyword Parameters

To use a parameter as a keyword parameter, you can specify the parameters in any sequence, but you must precede them by the keyword.

Positional Parameters

To use a parameter as a positional parameter, you do not need to include the keyword, but you must specify the parameters in the same order as in the statement syntax.

PROFILE Command

Each user profile begins with a PROFILE command and ends with an END PROFILE command. The PROFILE command provides global information about the processing options available to a particular user profile.

The syntax is:

PROFILE **PASSWORD name GROUP userid ID 'character string' PRINT BATCHING SUBFILES subfile name PAGES integer CALLS integer**

Note: *The PROFILE statement begins with the PROFILE command and can contain optional parameters. The parameters are not inherited by subordinate profiles.*

All of the following PROFILE command parameters are optional:

PASSWORD name

You can optionally specify a user password in the user profile as another level of security beyond the user ID.

- Make passwords from 1 to 8 alphanumeric characters. Make the first character alphabetic and do not use the system delimiter.
- If a profile contains a password, then include this password in the logon command.
- If the profile specifies a password and the user does not supply it, the logon is rejected.

GROUP userid

The GROUP parameter links profiles together in order to build a hierarchy of profiles.

A profile can only have one GROUP parameter followed by the user ID of another profile. The user ID specified becomes the parent from whom the profile being created inherits characteristics.

- If no GROUP parameter is present in a profile, VISION:Inform automatically groups the profile with the SYSTEM profile.
- If the user ID specified in the GROUP parameter does not exist in the foreground library, a diagnostic message is issued and the user is unable to log on to the system.

ID 'character string'

Use the ID parameter to add a string of information, up to 30 characters in length, to the banner page of every processed report. Enclose the ID information in single quotation marks. This parameter is not available to workstation client platforms.

PRINT

PRINT forces all reports by the user to the destination indicated in the INFREPT DD statement in the JCL for the Background Processor. That is, the user will not be able to route reports back to the terminal for viewing, regardless of the class under which the query runs. This parameter is not available with workstation client platforms.

Note: *Queries submitted from a user ID with a PRINT parameter are always treated as if the PRINT operand was supplied with the SUBMIT command. Report output is never stored on the communication file.*

BATCHING (Not Applicable to Immediate Response)

BATCHING specifies that all queries or tasks for this user are to be batched. Queries and tasks that cannot be batched are rejected.

If BATCHING is specified in a user's profile, VISION:Inform issues a warning message whenever the user enters a query statement that would prevent the query from being batched. For more information on query batching, see Chapter 4, "Controlling Processing".

SUBFILES subfile names (VISION:Bridge and Batch Simulator only)

SUBFILES specifies the names of the subfiles that this user profile is authorized to create and output. A complete discussion of subfile generation can be found in Chapter 4, "Controlling Processing".

PAGES integer (Immediate Response only)

PAGES specifies the number of output pages after which a checkpoint will occur. Make the value a positive integer. If the value is zero, no limit is imposed (no checkpoint will be taken). The default is zero.

CALLS integer (Immediate Response only)

CALLS specifies the number of database calls after which a checkpoint will occur. Make the value a positive integer. If it is zero, no limit is imposed (no checkpoint will be taken). The default is zero.

EXCLUDE Command

The EXCLUDE command restricts access to databases that would otherwise be inherited from a group profile. If the user attempts to access a database excluded in the profile, a diagnostic message is issued.

- In VISION:Bridge and Immediate Response, the message is issued as soon as the query statement is validated.
- In workstation client platforms, the message is issued when the query or task is ordered. Since users only have access to glossaries of databases included in their own or a parent profile, this situation occurs only when one of these profiles has been changed and the glossaries stored locally at the workstation have not been updated.

The syntax is:

EXCLUDE DATABASE database names

The parameter is:

DATABASE database names

DATABASE specifies the name of the databases being excluded from access. The database name is required, but the DATABASE keyword is optional.

INCLUDE Command

The INCLUDE command specifies the databases that you can access. The INCLUDE statement provides VISION:Inform with the database name and the processing classes the user can use.

The syntax is:

```
INCLUDE     DATABASE database name CLASS { integer | integer, integer }
            PRIVATE PSBNAME name
```

The command also specifies, through the PRIVATE parameter, whether that database can be inherited by subordinate profiles. One INCLUDE statement is required for each database to be accessed.

If the user attempts to access a database not defined in the profile, VISION:Inform attempts to satisfy the request by searching the parents of the profile. If this does not fulfill the request, VISION:Inform issues a diagnostic message.

Since VISION:Journey (remote platform) users only have access to glossaries of databases included in their own or a parent profile, this situation occurs only when one of these profiles has been changed and the queries referencing the obsolete glossary at the workstation have not been updated.

The parameters are:

DATABASE database name

DATABASE specifies the name of one database that the user can access. The database name is required, but the DATABASE keyword is optional.

CLASS integer or integer, integer

CLASS is an optional parameter of an INCLUDE statement. It specifies the class or range of classes this user ID can use when submitting a query or task that references this database. Because Background Processors can be programmed to process specific classes, you can use this parameter to control when a query or task is processed and by which Background Processor.

PRIVATE

The PRIVATE parameter prevents subordinate profiles from inheriting access to the database named within the INCLUDE group.

If a subordinate user ID does not have explicit access to a database, VISION:Inform searches the profile hierarchy looking for authorization at a higher level. If the database is defined as PRIVATE at a higher level, it is not available to this user. In this way, the PRIVATE parameter effectively prevents access to a database from being inherited by subordinate users.

PSBNAME name (CICS Immediate Response only)

The PSBNAME parameter specifies the Program Specification Block (PSB) VISION:Inform uses to access the database specified in this INCLUDE statement. If you omit this optional parameter, VISION:Inform uses a default PSB name from the PARMBLK.

RESTRICT Command

The RESTRICT command is a subcommand within an INCLUDE group. It restricts access to particular segments or fields within a database. Access is denied by specifying the desired segment or field names in a RESTRICT statement.

RESTRICT denies access to individual segments and fields for the user ID, as well as any subordinate user profiles. You can have any number of RESTRICT statements within an INCLUDE group. RESTRICT statements must appear on separate lines from the INCLUDE statements.

For information on the use of both RESTRICT and SELECT, see the section Using the RESTRICT and SELECT Commands Together.

The syntax is:

RESTRICT SEGMENT segment names FIELD field names

The parameters are:

SEGMENT segment names

The SEGMENT parameter specifies database segments that cannot be accessed by this user ID.

FIELD field names

The FIELD parameter specifies database fields that cannot be accessed by this user ID.

SELECT Command

The SELECT command is a subcommand within an INCLUDE group providing database access at the record level using conditional field selection logic. It is automatically and transparently included in all tasks and queries submitted by the user against the database specified in the INCLUDE statement.

As with the RESTRICT subcommand, the SELECT is inherited by any subordinate user ID subject to this profile. Place the RESTRICT command on a separate line from the INCLUDE statement.

Using the RESTRICT and SELECT Commands Together

Take care when using the RESTRICT command with the SELECT command. For example, assume the following statements are entered in a profile (one statement could be inherited from a higher level profile):

```
RESTRICT SEGMENT PLANT
SELECT IF PLANTKEY = 20150      (PLANTKEY is in the PLANT segment)
```

These statements are accepted as valid. However, since PLANTKEY is in the PLANT segment (which is restricted), all queries and tasks against the database are rejected when they are submitted.

Note: *The field name used on the left side of the logical expression must specify the root key of the database*.

The syntax is:

SELECT IF logical expression

The parameter is:

IF logical expression

The IF parameter and a logical expression are required. It specifies the conditions under which access to the database is permitted. Any logical expression will be either true or false. The query can access the data when the logical expression is evaluated as true.

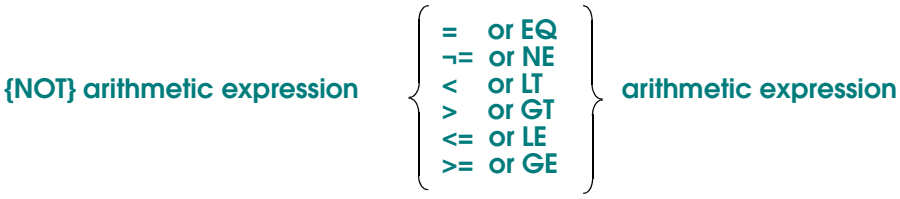
The logical expression can be a simple expression or consist of a nesting of relational and arithmetic expressions.

An example of a simple expression is:

```
IF DEPT NE 'SALES'
```

Relational Operators

A relational expression consists of two arithmetic expressions connected by a relational operator:

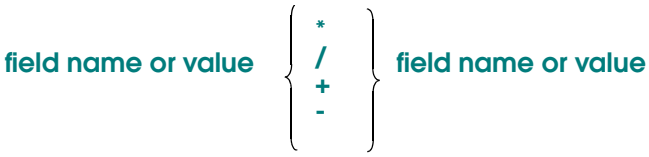


The relation can also be preceded by the logical operator NOT. For example:

```
IF NOT DEPT = 'SALES'
```

Arithmetic Operators

An arithmetic expression can be simple or consist of field names or values connected by arithmetic operators:



TO Operator

TO is a special operator used to designate a range of values. The expression IF SALARY + BONUS EQ 20000 TO 50000 is interpreted by VISION:Inform to mean:

```
IF (SALARY + BONUS GE 20000) AND (SALARY + BONUS LE 50000)
```

Lists in Relational Expressions

You can use relational expressions to test a field to see if it is one of several values. Use the EQ and NE operators:

- EQ (=) implies OR, that is, the field is equal to at least one of the items in the list.
- NE (¬=) implies AND, that is, the field is not equal to any of the items in the list.

Any item in the list can be a range of values.

Example 1

```
IF QUANTITY = 20   30   40
```

This relational expression is true whenever QUANTITY is equal to 20, to 30, or to 40.

Example 2

```
IF QUANTITY NE 20   30   40
```

The relational expression is true whenever QUANTITY is not equal to any of the values 20, 30, or 40.

Example 3

```
IF QUANTITY = 20, 30 TO 40, 65
```

This expression tests for two specific values, as well as a range of values. The expression will be true if QUANTITY is equal to 20, or 65, or falls in the range from 30 to 40.

Multiple Relational Expressions

The preceding logical expressions consist of only one relational expression. A logical expression can also consist of two or more relational expressions connected by the binary logical operators OR or AND:

relational expression { **OR** / **AND** } **relational expression** { **OR** / **AND** } **...**

For example:

IF SALARY = 20000 (relational expression) AND DEPT NE 'SALES' (relational expression)

(logical expression)

A logical expression can also be preceded by the selective operator ANY, ALL, or NOT.

An example of a command showing all expression types is:

SELECT IF SALARY + BONUS (arithmetic expression) EQ 20000 TO 50000 AND DEPT NE 'SALES'

Operator Precedence

The operators in a logical expression are processed in the following order:

*, \
+,-
EQ, NE, LT, GE, LE, GT, TO
NOT
AND
OR

Operators at the same level are processed from left to right. You can use parentheses to override the processing order. If parentheses are nested, the innermost parenthetical expression is evaluated first.

END INCLUDE Command

END INCLUDE is a required command used in conjunction with the INCLUDE commands. Each INCLUDE group begins with an INCLUDE command and must end with an END INCLUDE command.

PROHIBIT Command

The PROHIBIT command prohibits the use of either an entire command or certain keywords of a command. The users whose profile contains a PROHIBIT command cannot use the command or command keywords specified.

- Use a separate PROHIBIT command for each command to be prohibited.
- If any keywords for a particular command are to be prohibited, place them on the same PROHIBIT statement.
- Use this command only with VISION:Bridge or Immediate Response.

Note: *LOGON and QUIT cannot be prohibited.*

Use the PROHIBIT command only to prohibit use of particular command input, query language statements, and keywords.

Primary commands, entered on the Command line of a panel, cannot be prohibited. Primary commands are listed in the CUSTOM macro which is part of PARMBLK.

The syntax is:

PROHIBIT COMMAND command KEYWORD keyword

The parameters are:

COMMAND command

The COMMAND parameter specifies the command to be prohibited. A command is required, but the COMMAND keyword is not.

KEYWORD keyword

The KEYWORD parameter specifies the keyword or keywords to be prohibited. It is optional, and if it is not specified, the entire command is prohibited.

END PROFILE Command

The END PROFILE command is required. Place it as the last statement in a user profile. The END PROFILE command tells VISION:Inform that the end of a profile definition has been reached.

Sample User Profile

The following shows a sample user profile.

```
1 - PROFILE PASSWORD GEM GROUP PRSNL

2 - EXCLUDE DATABASE INVNTRY

3 - INCLUDE DATABASE PAYROLL CLASS 11 PRIVATE

4 - RESTRICT SEGMENT S1 S2 FIELD F5 F7 F11

5 - SELECT IF ROOTKEY GT 10000 AND ROOTKEY LT 20000

6 - END INCLUDE

7 - END PROFILE
```

Figure 3-1 Sample User Profile

Each line of the profile is explained in the following table.

Line no.	Explanation
1	Make the PROFILE command the first command you enter when creating a user profile. The command is followed by a number of parameters and their key values. The password GEM controls this user's access to VISION:Inform. The GROUP parameter specifies that the profile is subordinate to a profile called PRSNL from which it can inherit profile characteristics.
2	This EXCLUDE command denies this user ID access to the database INVNTRY. The statement implies INVNTRY has been included in the group level profile PRSNL to which this profile is subordinate.
3	This INCLUDE command is the first command in an INCLUDE group. The PRIVATE parameter of this INCLUDE command means this user ID can access the database PAYROLL without allowing subordinate profiles to inherit access. The CLASS parameter is used here to specify class 11 for all tasks and queries this user submits for the PAYROLL database.
4	The RESTRICT command prevents access to segments S1 and S2 and fields F5, F7, and F11 from database PAYROLL for this user ID. Notice that this is part of an INCLUDE group, but is not an parameter of the INCLUDE command. RESTRICT is a unique command with its own parameters.
5	Like RESTRICT, the SELECT command is a unique command used only within an INCLUDE group. This statement is transparently appended to all requests made by this user ID against the PAYROLL database. It limits access to records in which the value of field ROOTKEY, the database root segment key field, is between 10000 and 20000.
6	This required command ends the INCLUDE group for the database PAYROLL.
7	This required command ends the PROFILE group.

Structuring Profile Hierarchies

Use the VISION:Inform security commands to build a profile hierarchy with any degree of complexity. This section provides a number of examples ranging from very simple to moderately complex.

Inherited and restricted structures are only conceptual distinctions. You can use the security commands to build a profile hierarchy that combines characteristics of both types of structures. When developing a profile hierarchy, consider who requires access to what data, the ease of maintenance, and the clarity of function.

Single-Level Structures

Single-level structures consist of one or more user profiles that are not tied to a group profile through the use of the GROUP parameter in the PROFILE statement. Each profile is completely self-contained with access to databases, processing options, and restrictions to data explicitly detailed in each profile. They are all, by default, subordinate to the SYSTEM profile.

Assume an installation with three databases called PAYROLL, PERSONEL, and INVNTRY. Each database is used primarily by a different department. The following single-level structure could be used.

Note: *PAYROLL, PERSONEL, and INVNTRY are sample database names.*

USER ID USER1

```
PROFILE PASSWORD ONE
INCLUDE PAYROLL
END INCLUDE
```

USER ID USER2

```
PROFILE PASSWORD TWO
INCLUDE PERSONEL
END INCLUDE
```

USER ID USER3

```
PROFILE PASSWORD THREE
INCLUDE INVNTRY
END INCLUDE
```

Each profile explicitly includes one database. This is the minimum information needed to define a user profile. Each profile has explicit access to one and only one database.

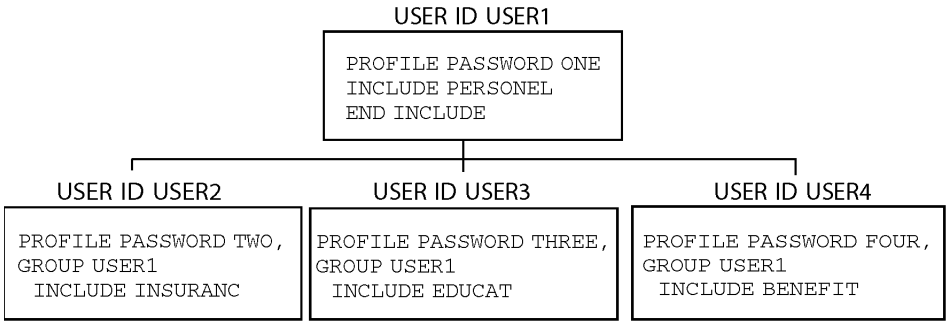
PROFILE	DATABASES AVAILABLE
USER1	PAYROLL
USER2	PERSONEL
USER3	INVNTRY

If the SYSTEM profile contains any INCLUDE commands, specify the profiles as PRIVATE to deny these user IDs access to them. Profile names are not associated with a profile until the profile has been stored in the foreground library using the SAVE command.

Two-Level Inherited Hierarchies

This two-level inherited hierarchy example introduces a group profile with three subordinate profiles, each of which can inherit access to data from the higher level profile. Of course, you can have any number of subordinate profiles.

Note: *PERSONEL, INSURANC, EDUCAT, and BENEFIT are sample database names.*



This structure places USER1 at the top of the profile hierarchy.

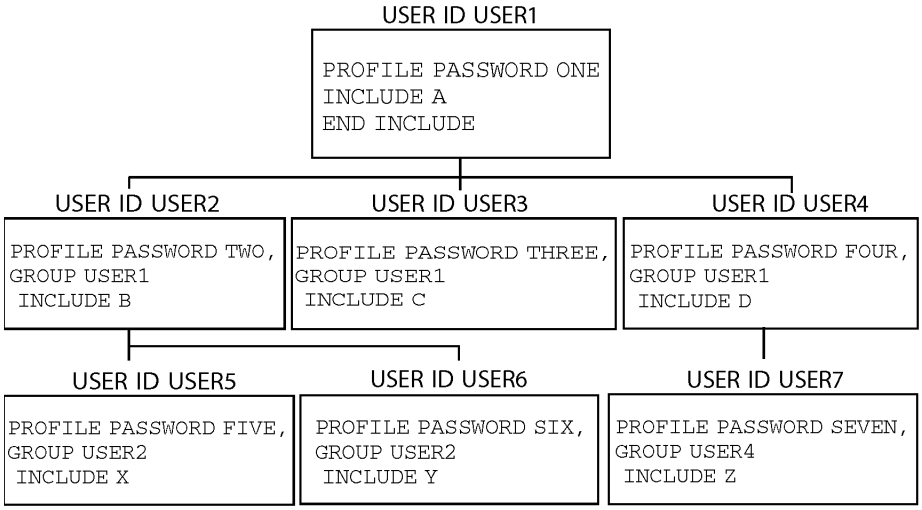
- It assumes that any INCLUDE statements in the SYSTEM profile contain the PRIVATE specification.
- It is a characteristic of inherited structures that each subordinate profile, linked to USER1 by the GROUP parameter, inherits access to the databases available to USER1.

In addition, each subordinate profile has access to those databases listed within it. Thus, USER2 gets access to the PERSONEL and INSURANC databases. USER3 gets access to the PERSONEL and EDUCAT databases. USER4 gets access to the PERSONEL and BENEFIT databases. Characteristics detailed in the profile for USER2 have no impact on those listed in profiles for USER3 or USER4.

Multi-Level Inherited Hierarchies

The multi-level inherited hierarchy structure is applicable where certain departments or projects require access to particular databases and individuals within the departments or projects require access to additional databases. Assume any INCLUDE statements in the SYSTEM profile contain the PRIVATE parameter.

Note: *A, B, C, D, X, Y, and Z are sample database names*.



In this example, the profile called USER1 is a generic profile for everyone. Access to database A is available to all subordinate users through inheritance of characteristics. Second-level profiles are linked directly to USER1 by the GROUP parameter. Third-level profiles are linked indirectly through their own group profiles.

PROFILE	DATABASES AVAILABLE
USER1	A
USER2	A, B
USER3	A, C
USER4	A, D
USER5	A, B, X
USER6	A, B, Y
USER7	A, D, Z

You can easily maintain and update inherited structures by introducing global changes in the upper levels of the structure. Notice also that the lowest levels in the hierarchy possess the greatest power in terms of access to various databases.

Restricted Hierarchies

There are cases where the accumulative effect of an inherited hierarchy is not appropriate. For example, placing managers at the top of the hierarchy, giving them access to the most databases, and imposing the least restrictive processing characteristics. You would give subordinate profiles access to data on a need-to-know basis. This situation is descriptive of a restricted structure.

You can approach restricted structures in the following ways:

- You can start by defining a high-level user profile with access to virtually all the data in the system. Then you can selectively restrict access by subordinate profiles through the use of the EXCLUDE command.
- Or, you can define the same highest level profile with access to everything. Then, use the INCLUDE command and the PRIVATE parameter to selectively inhibit the inheritance of certain databases by subordinate profiles.

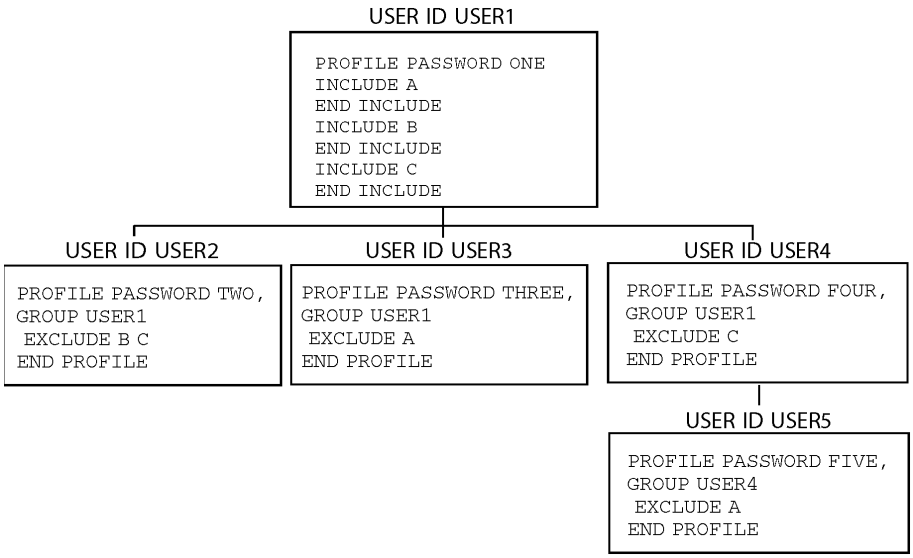
The primary consideration when choosing between the two approaches should be the nature of the security requirements at your installation.

- If you have a situation where access to databases is relatively unrestricted, it will be easier for you to build and maintain a hierarchy based on the exclusion approach.
- However, if most of the data in your system is restricted, it will be easier for you to build a hierarchy that takes advantage of the INCLUDE command and PRIVATE parameter. Both approaches are discussed in the following subsections.

Restriction Through Exclusion

The restriction through exclusion approach defines an all-inclusive user profile. Subordinate profiles inherit access to a subset of the various databases by selective restriction through use of the EXCLUDE command. In this situation, you can use the SYSTEM profile instead of USER1.

Note: *A, B, and C are sample database names.*

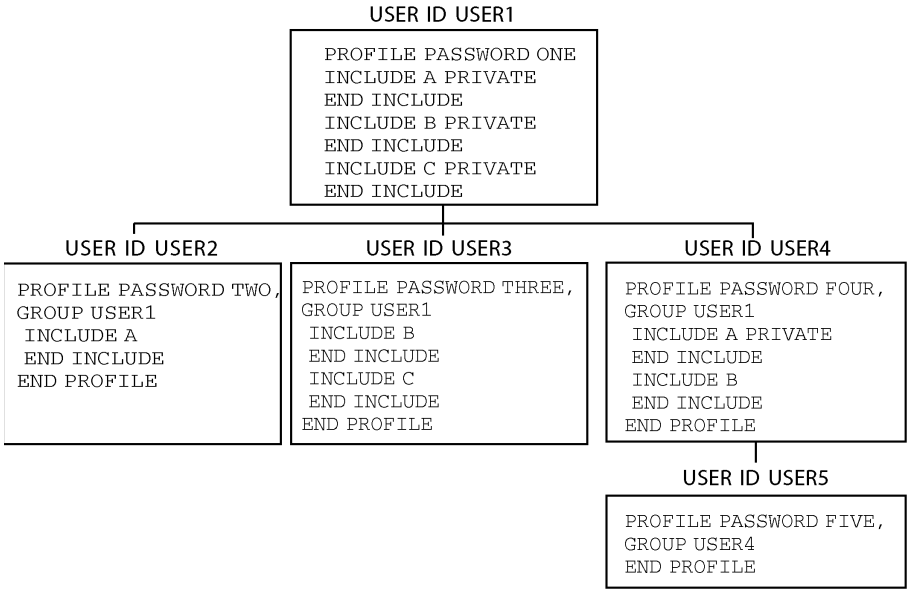


Note that USER1 has access to databases A, B, and C explicitly defined. Use the EXCLUDE command to selectively restrict the access of subordinate users. In this way, lower levels in the hierarchy generally have access to fewer databases than higher levels.

Restriction Through Inclusion

The restriction through inclusion approach uses the INCLUDE command and the PRIVATE parameter. In this situation, you can use the SYSTEM profile instead of USER1.

Note: *A, B, and C are sample database names.*



The restriction through exclusion and restriction through inclusion examples are designed to build profiles with access to exactly the same databases.

PROFILE	DATABASES AVAILABLE
USER1	A, B, C
USER2	A
USER3	B, C
USER4	A, B
USER5	B

Notice that the inclusion approach requires more statements than the exclusion approach. While this might not seem significant in this context, it can become a burden in large installations with many user profiles to maintain. Again, the approach you take depends on the way you want to structure the hierarchy.

User Profile Commands and Parameters Summary

Note: *The parameters on the PROFILE statement are not inherited by subordinate profiles.*

This section summarizes the PROFILE, PROHIBIT, EXCLUDE, INCLUDE, RESTRICT, SELECT, END INCLUDE, and END PROFILE commands.

PROFILE	A required command indicating the beginning of a profile definition.
PASSWORD name	Specifies the password required to access this user profile.
GROUP userid	Specifies the name of the PROFILE group to which this user ID belongs. The user ID specified must exist in the foreground library. Default is GROUP SYSTEM.
ID 'character string'	Specifies identification information to be printed on the processed query headers. Maximum length of 30 characters. Do not use with workstation client platforms.
PRINT	Specifies that all query output is to be forced to the destination specified for INFREPT in the Background Processor job. Do not use with workstation client platforms.
BATCHING	Specifies queries and tasks MUST be batched when submitted. Queries and tasks that cannot be batched are rejected. Do not use with Immediate Response.
SUBFILES subfile names	Specifies the names of the subfiles this user ID can output. Use with VISION:Bridge and Batch Simulator only.

Note: *When you specify PAGES 0 or CALLS 0, the Immediate Response Processor does not take checkpoints. If these are omitted, the default is zero.*

PAGES integer	Specifies the number of output pages after which a checkpoint will occur. Make the integer positive. If it is zero, no limit is imposed. The default is zero. Use with Immediate Response only.

CALLS integer	Specifies the number of database calls after which a checkpoint will occur. Make the integer positive. If it is zero, no limit is imposed. The default is zero. Use with Immediate Response only.

Note: *You cannot use PROHIBIT to prohibit 3270 panel primary commands.*

PROHIBIT	An optional command to prohibit the use of either an entire command or certain keywords of a command. The users of profiles subordinate to a profile with a PROHIBIT command cannot use the command or command keywords specified. Do not use with workstation client platforms.
COMMAND command	A required parameter of the PROHIBIT command that specifies the command to be prohibited. Specify only one command on each PROHIBIT statement.
KEYWORD keyword	An optional parameter that specifies the keywords to be prohibited; if it is not specified the entire command is prohibited. Place all keywords for a particular command on the same PROHIBIT.
EXCLUDE	An optional command that specifies that a database defined for access in a parent's user profile cannot be accessed by this user ID.
DATABASE database names	A required parameter of the EXCLUDE command that specifies the name of the databases that this user ID cannot access. You can specify more than one database name.
INCLUDE	An optional command that specifies the databases that can be accessed by this user ID. Specify a separate INCLUDE command for each database to be accessed.
DATABASE database name	A required parameter of the INCLUDE command that specifies the name of the database that this user ID can access.

CLASS integer or integer, integer	An optional parameter of the INCLUDE command that specifies the class or classes that this user ID can specify on the Submit panel or SUBMIT statement. ■ If you do not specify CLASS during a submit, VISION:Inform defaults to the lowest value specified in the profile. ■ If you do not specify CLASS in the subordinate profile, VISION:Inform defaults to the INCLUDE for this database parent profile and then the PARMBLK specification. ■ If you do not specify CLASS during a submit, or if one is not supplied from the profile hierarchy, then VISION:Inform uses the default submit class specified in the PARMBLK DEFCLAS parameter.
PRIVATE	An optional parameter of the INCLUDE command that specifies that the included database is accessible to this user ID only and cannot be inherited by subordinate profiles.
PSBNAME	An optional parameter that specifies the Program Specifications Block (PSB) to be used to access the database specified above. If you omit PSBNAME, a default PSB name from the PARMBLK is used. CICS Immediate Response only.
RESTRICT	An optional command used within an INCLUDE group to specify restricted items for a database. You can use multiple RESTRICT statements.
SEGMENT segment names	An optional parameter of the RESTRICT command that specifies the database segments that cannot be accessed by this user ID. You must define each of these segments in a file definition to use this parameter.
FIELD field names	An optional parameter of the RESTRICT command that specifies database fields that cannot be accessed by this user ID.

SELECT

An optional command used within an INCLUDE group to specify selection criteria for limiting data access at the root level. The SELECT statement is added to all queries or tasks against the database specified. The user entering the query or task will be unaware of the expression being added.

IF logical expression

A required parameter of the SELECT command that specifies the selection criteria to be used for limiting access to the database.

- The logical expression can reference root and flag fields only.
- It can contain a maximum 790 characters.
- Indicate continuation lines by commas.

END INCLUDE

Signifies the end of an INCLUDE group. Required entry for each INCLUDE command used in a profile, except the last, although it is recommended for the final entry as well.

END PROFILE

A required command that signifies the end of a profile definition.

User Exit Routines

VISION:Inform provides two exit facilities. The profile exit provides you the ability to dynamically change user profile values during a VISION:Inform session.

The INFREPT exit (which is not applicable to Immediate Response) provides you the means to gain control over the processing of the INFREPT data set. The user routines are coded using standard IBM linkage conventions to interface with the exits.

Profile Exit

With the profile exit routine you can:

- Supply a database SELECT statement for a database that does not have one.
- Override an existing database SELECT statement.
- Cancel a database SELECT statement.
- Supply ID information for reports.
- Interface with security packages for password processing.

At log on time, if a profile exit routine exists, it is invoked once for each database defined in the user's profile or inherited from parent profiles. The exit is also invoked once at session termination (when the QUIT command is entered) for any housekeeping functions required in the user's routine.

- If a SELECT or an ID statement is supplied by the profile exit routine, the SELECT or ID in the user profile is overridden.
- If the exit routine does not supply SELECT or ID information, the existing profile values are used.

The SELECT statement provided by the exit is validated when the user submits a query against the database. A diagnostic message is issued at that time if the SELECT statement is incorrect.

Any values provided by the profile exit routine, whether they are modifications or additions to the user profile, apply only to the session. They are not permanent changes to the user profile.

INFREPT Exit

With the INFREPT exit routine, you can have complete control over the processing of the INFREPT data set during Background Processor execution.

The INFREPT exit routine is invoked by VISION:Inform just before the INFREPT data set is opened. At this point, your routine sets indicators which tell VISION:Inform at what points during INFREPT processing you want to gain control. You can alter the point of control repeatedly during the processing of the INFREPT data set and one final time just before the INFREPT data set is closed.

For a detailed explanation on how to code and install a user written exit routine, see VISION:Inform *Installation Guide* for your environment.

Chapter

4

Controlling Processing

VISION:Inform provides you with the means to control and monitor various processes. This chapter describes:

- Batching and Query Processing.
- Controlling Background Processors.
- Commands for Controlling Resources.
- Techniques for Controlling Resources.
- Distributing Report Output.
- Alternate Report Output Formats.
- Understanding PCB Selection.
- Understanding Database Calls.
- Using Subfiles (VISION:Bridge and Batch Simulator Only).
- Calling External Routines.
- Controlling Storage.
- Controlling Foreground Library and Communication File Space Utilization.
- Using VISION:Inform with DB2 (Not Applicable to Immediate Response).
- Logging Background Processor Activity.
- Optimizing Database Access.

Immediate Response Queries

Note that **Immediate Response queries** (queries processed by the Immediate Response Processor) are not batched and are always executed immediately by the Foreground Processor.

- Immediate Response queries do not use a Background Processor or a communication file and do not have a processing class.
- The output from an Immediate Response query always automatically returns to the originating terminal.
- Hard-copy output from Immediate Response queries is obtained by running the Immediate Response query with the Batch Simulator.

Batching and Query Processing

VISION:Inform accumulates queries and tasks in the communication file for processing by Background Processors. The queries and tasks remain in the communication file until a Background Processor becomes available to process them.

The Background Processor processes queries and tasks according to their class. When you submit a task or query, VISION:Inform assigns a class. Background Processor control statements specify batching criteria.

Batching

The term "batching" refers to how a Background Processor places queries and tasks into groups, or batches, that access the same database or logical data view. The entire batch is processed against the database or logical data view in a single pass. You specify the conditions for batching.

Batching is not always suitable for all users.

- Some users need the results of their queries and tasks sooner than the time it takes to accumulate the specified minimum number of queries to batch together.
- Other users do not need to see their results immediately, and hence, prefer batching and its efficient use of resources.

VISION:Inform provides you the flexibility to satisfy the needs of both types of users.

The following sections describe the role that the various VISION:Inform components play in the batching of queries and tasks.

Queries and Tasks in the Communication File

The communication file contains the queries and tasks awaiting processing. Queries and tasks remain there until a Background Processor becomes available and selects them for processing.

For information regarding the creation of the communication file, see the VISION:Inform *Installation Guide* for your environment.

Processing Classes

Notes: *For specific information, see:*

- *DATABASE Command*,
- *Submit Panel and SUBMIT Command*, *and*
- *SUBMIT Command (Edit Mode and System Mode)*.

Specifying Classes for Background Processors

Class specification determines how queries and tasks are batched and processed. You determine the class according to your requirements.

- You indicate the class or range of classes for a Background Processor on the DATABASE statement.
- You specify a class with an integer from 1 to 14 in the DATABASE command CLASS parameter.

The Background Processor processes queries or queries by database and by class.

Specifying Class for a Query or Task

When you submit a query or order a task, the class is obtained from one of the following:

- The Submit panel or the Order screen (VISION:Bridge and workstation clients)
  or The CLASS parameter of the SUBMIT command (Batch Simulator)
- The profile hierarchy.
- A default in PARMBLK. The PARMBLK default is used when neither the user profile nor its parent profiles contain a class and no class is specified at submit time. The relevant PARMBLK keyword is DEFCLASS, which has a default value of 10.

Queries and tasks are routed by the Foreground Processor to the communication file where they are held until a Background Processor for the particular class and database becomes active.

The report output from the task or query normally returns to the communication file, unless the SUBMIT for the query specifies the PRINT parameter, in which case, report output is directed to the destination indicated by the Background Processor INFREPT DD statement. Alternate output formats, specified in the query or at SUBMIT time, allow users to create web-enabled data formats from queries. These formats include HTML, comma-deliminted, tab-delimited, and plain text. These alternate output formats are not stored on the communication file, but are instead written to dynamically allocated files by a Background Processor.

Query Submission Flow

Figure 4-1 illustrates the possible paths queries and tasks can take in the VISION:Inform environment. In the diagram, solid lines represent tasks or queries and broken lines represent reports. This figure does not apply to Immediate Response queries.

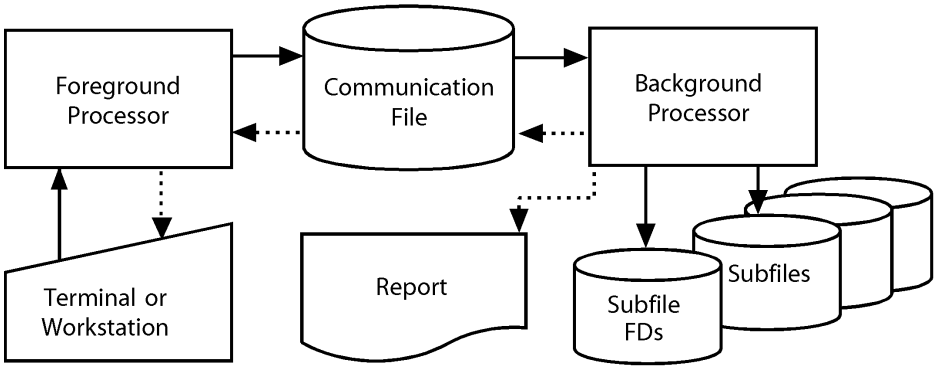


Figure 4-1 Query Path

The User Profile Function

Whenever a user submits a task or query, VISION:Inform checks the user profile for a class specification.

- If it finds a class specification, it verifies the class.
- If it does not find one, it searches the parent profiles before assigning the default class from the PARMBLK.

It also searches the profile to see if the user's query or task can be batched and to see if the user can access the specified database.

Queries or tasks are rejected from submission if the user does not have:

- Access to the specified database or class.
- Access to a database field, segment, or record from which the profile restricts access.

Batching Restrictions

The batching facilities of VISION:Inform are optional. The BATCHING parameter and CLASS parameter in the user's profile specifies the way queries and tasks are to be processed. Classes are specified as integers ranging from 1 through 14. The Background Processor processes a given class of queries as specified by the input parameter data set, INFIN.

A query or task will not be batched if:

- It modifies a database or system field with a LET statement. These queries can be made batchable and still obtain the same results by modifying temporary fields instead of the flag or database field.
- It uses a temporary field in the denominator of a PCT or RATIO statement.

If either condition is attempted and the BATCHING parameter is present in the user's profile, the query is rejected when submitted.

If the class specified in the Submit panel or SUBMIT command is not within the range of classes specified in the user profile or the parent profile, the submit is rejected.

Queries that are executable but not batchable are processed separately by the Background Processor. If a query is found to be not executable, a diagnostic message is issued when the user attempts to submit the query.

The Background Processor

You control the way the Background Processor operates through commands and parameters. The parameters you specify are input to each Background Processor when it executes. JCL examples for executing the Background Processor are supplied for your environment on the installation tape.

Commands and parameters specify the following:

- The class or classes to process.
- Whether to batch queries and tasks.
- The maximum and minimum number of queries and tasks to batch in one pass of the database.
- The databases to process.
- The time interval to check the communication file for queries and tasks awaiting processing.
- Information to control Background Processor logging.
- When to automatically terminate the Background Processor.
- Control of the communication file query/task queue and print queue.
- The disposition of any HELD status queries.
- DB2 connection information.
- Checkpoint restart information (IMS only).
- System date override information.
- Run mode specification (24-bit/31-bit).

Controlling Background Processors

The Background Processors accept processing options through the commands and parameters you specify in the INFIN data set. The Background Processors use these commands and parameters from the INFIN data set to control processing of queries and tasks.

The INFIN data set must contain a CONTROL statement with a NAME parameter. Other commands are optional.

Statement Format

You specify the commands and parameters in the form of 80-character statements. Positions 1-72 contain the commands and parameters. Positions 73-80 contain optional sequence numbers. At least one parameter for a command must be included on the line with the command statement.

Continuation lines and comments:

If the last non-blank character in a line is a comma, the next line is treated as a continuation.

A semi-colon (;) signifies that the balance of the line is to be treated as a comment.

Commas

You can add commas anywhere in a statement to improve readability. Commas are not required.

Background Processor Error Processing

If the Background Processor encounters errors opening the INFIN data set or encounters invalid statements, the Background Processor reacts as specified in the PARMBLK option ERROPT. See the VISION:Inform *Installation Guide* for your environment.

- The default for the ERROPT option is that processing will continue with automatically supplied defaults for the affected statements. (ERROPT=CONT)
- At end of job, the Background Processor passes a return code to the operating system. For details, see VISION:Inform *Messages and Codes*.
- The Background Processor writes appropriate diagnostic messages to the sequential log file and to the status log on the communication file.

You can also specify the ERROPT option to terminate processing when encountering INFIN problems. The Background Processor will then quit and pass a return code of 16 to the operating system. (ERROPT=TERM)

The following sections describe the Background Processor commands and parameters.

CONTROL Command

The CONTROL statement specifies how the Background Processor processes queries and tasks. The CONTROL command is required.

Note: *You can add commas anywhere in a statement to improve readability. Commas are not required.*

The syntax is:

```
CONTROL    NAME name, QTIME integer, MAXTIME integer,
           MAXQRY integer, MINQRY integer, PRINT, SYSDATE date
```

The parameters are:

NAME name

A required parameter that specifies the name of the Background Processor. This name is used for uniquely identifying the Background Processor in log records and message displays, and terminating the Background Processor using the Foreground Processor.

QTIME integer

An optional parameter that specifies the minimum time interval (in minutes) for checking the queue for tasks or queries to be processed.

- If you do not specify this parameter, the Background Processor checks the queues once for each set of batchable and non-batchable queries for each database specified and terminates.
- If you specify QTIME, the Background Processor terminates only if MAXTIME is exceeded or the Background Processor is terminated with the TERM command.

MAXTIME integer

An optional parameter that specifies the maximum time (in minutes) the Background Processor is active. The Background Processor automatically terminates when the specified time interval is reached even if the current queries and tasks being processed are not finished.

- If you do not specify the MAXTIME parameter, the Background Processor continues to process everything queued and checks for more tasks or queries at QTIME intervals until it is shut down by the TERM command.
- If the performance option (OPTION SWITCH1 ON) is used, the MAXTIME shutdown may not be honored as soon as it would be when the performance option is not in use.

MAXQRY integer

An optional parameter that specifies the maximum number of tasks or queries to be batched together for one pass of the database. If you do not specify this parameter, the default value comes from PARMBLK.

MINQRY integer

An optional parameter that specifies the minimum number of tasks or queries to be batched together for one pass of the database. If you do not specify this parameter, the default value comes from PARMBLK.

Note: *See the PARMBLK macro BGPRINT keyword description in the* VISION:Inform *Installation Guide for your environment*.

PRINT

An optional parameter that specifies that this Background Processor can only retrieve and print reports that are routed to print from an online session or the Batch Simulator. Output is directed to the INFREPT data set. If DATABASE statements follow, they are ignored. No database processing is executed by a PRINT Batch Processor.

SYSDATE date

An optional parameter that specifies an override for the system date to be used for a particular Background Processor execution. Specify the date in MMDDYYYY format.

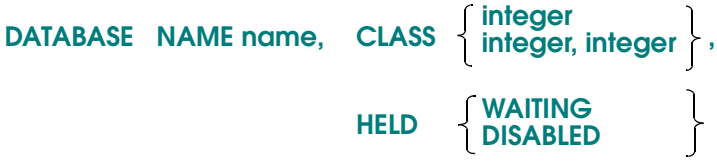
For testing in a future year environment, specify the SYSDATE keyword and specify a date in the future, overriding the system date.

In addition, VISION:Inform provides access to the new 21st century date flag fields, JULANX, LILIAN, and TODAYX.

DATABASE Command

An optional command that specifies which tasks or queries to process against which database. If you do not specify DATABASE, all databases are processed. Any database can be specified. You can specify multiple DATABASE statements.

The syntax is:



The parameters are:

NAME name

A required parameter specifying a database for the Background Processor to select for processing. The specified database definition must have already been promoted to the foreground and background libraries.

CLASS integer or integer, integer

An optional parameter that specifies the class or range of classes the Background Processor can process for the defined database. Class ranges are from 1 through 14.
If you do not specify CLASS, classes 1 through 14 are processed. Using a comma between two integers indicates a range.

An optional parameter that specifies how to set the status of queries for this database that are in HELD status during the start up of the Background Processor. Queries are set to HELD status when a database or Background Processor problem occurs.

- Specifying AWAITING resets the status to AWAITING for all queries that were put in HELD status due to a problem that occurred in a prior run of the Background Processor with the same name. These queries are now candidates to be processed by this Background Processor execution.
- Specifying DISABLED resets the HELD queries to DISABLED status until they are set to an AWAITING status by a Foreground Processor or Batch Simulator ENABLE command.

DB2 Command

An optional command that supplies the names required to process queries that access DB2 tables. This command is operational only if your VISION:Inform contains the relational support (DB2) option.

The syntax is:

DB2 SYSTEM name, PLAN name

The parameters are:

SYSTEM name

A required parameter that specifies the DB2 system name.

PLAN name

A required parameter that specifies the DB2 plan name.

Log Record Commands

The log record commands are BS00, BT00, BE00, CE00, DB00, IM00, LG00, QE00, QS00, and QT00.

These optional commands cause the specified records to be written to the sequential log file and the communication file status log as specified by the TARGET parameter.

After the Background Processor terminates processing, this information remains in the communication file until a Background Processor with the same CONTROL name is started. For a description of these files, see the section Logging Background Processor Activity.

TARGET is a parameter for the log record commands, BS00, BT00, BE00, CE00, DB00, IM00, LG00, QE00, QS00, and QT00.

WRAPNO is a parameter for only the QS00 log record command.

TARGET	{ **LOG** **STAT** }	An optional parameter that specifies whether the specific record type will be written to the sequential log file (LOG) and the communication file status log (STAT). Used with all the commands above, except that IM00 cannot be written to the communication file.
WRAPNO	**nnnn**	An optional parameter used only with the QS00 command that specifies the number of query started records to be maintained on the communication file status log; nnnn must be at least as large as MAXQRY in the CONTROL command. If WRAPNO is not specified, QS00 records will be stored sequentially as long as there is room for them.

BS00 Command

Optional command to write the Background Processor startup record.

BT00 Command

Optional command to write the Background Processor termination record.
Do not specify this record with a TARGET of STAT unless BS00 is also specified.

BE00 Command

Optional command to write Background Processor error records. Do not specify this record with a TARGET of STAT unless BS00 is also specified.

CE00 Command

Optional command to write control statement error records.

DB00 Command

Optional command to write database records.

IM00 Command

Optional command to write informational message records.
Do not specify this record with a TARGET of STAT.

LG00 Command

Optional command to write logging control records.

QS00 Command

Optional command to write query started records.

QT00 Command

Optional command to write query completion records.
Do not specify these records with a TARGET of STAT unless QS00 is also specified.

QE00 Command

Optional command to write query error records.
Do not specify these records with a TARGET of STAT unless QS00 is also specified.

OPTION Command

Optional. Specifies the method of running the Background Processor.

The syntax is:

The parameters are:

SWITCH1 { OFF | ON }

An optional parameter. You can use OPTION SWITCH1 to control performance. The default is SWITCH1 OFF.

OPTION SWITCH1 ON in the control statements selects the "speed" (versus stability) method of running the Background Processor. With the "speed" option the Background Processor will not be able to continue to execute if an unexpected problem occurs during data selection.

When the OPTION SWITCH1 ON control statement is specified for a Background Processor, the MAXTIME shutdown may not be honored as soon as it would have when the performance option is not in use (no OPTION SWITCH1 ON control statement).

ALTSORT { **OFF** / **ON** } An optional parameter that causes the Background Processor to operate at sort input and output exits. The default is ALTSORT OFF.

Note that some sorts, including SYNCSORT and PLSORT, do not operate with ALTSORT OFF.

MODE31 Command

Optional. Specifies the run-mode for a particular Background Processor run. This feature allows for users with special run-time run-mode considerations (such as CALL statements, User I/O, GDBI, etc.) to affect the 31-bit/24-bit run-mode control. The default run-mode for the background processor is to run in 31-bit execution.

The syntax is:

MODE31 SETTING { **OFF** / **ON** }

There is no default setting. This means in the absence of a MODE31 control statement, the Background Processor will not generate any 31-bit control parameters, defaulting to whatever has been set in M4PARAMS (default of 31-bit=yes) or in a Logical Dataview. You may specify the MODE31 keyword to override the setting from M4PARAMS or a Logical Dataview. The only reason you need to specify MODE31 is if you are using a CALL Query Language statement in a query, User I/O, or GDBI routines which cannot execute in 31-bit mode, and requires 24-bit execution.

The SETTING keyword for the MODE31 Statement is optional. OFF or ON must be specified if the MODE31 command is used.

The parameters are:

SETTING { **OFF** / **ON** } OFF causes the Background Processor to run in 24-bit mode. ON causes it to run in 31-bit mode.

Note: *In most circumstances, it is not necessary to use this control statement. If you have any questions about using this control statement, please contact Technical Support.*

Background Processor Command Examples

This section contains examples of the Background Processor input commands and parameters. Explanations follow each example.

Note: *You can add commas anywhere in a statement to improve readability. Commas are not required.*

Example 1

```
1       CONTROL NAME L, QTIME 30, MAXQRY 10
2           DATABASE NAME A  CLASS 4, 8
3           DATABASE NAME B  CLASS 8
```

Line no.	Explanation of statement
1	This statement specifies the input parameters for Background Processor L. The queue is checked for queries and tasks a minimum of every 30 minutes. The maximum number of queries and tasks included in each batch is 10.
2	This statement specifies that queries and tasks from classes 4 through 8 are processed by this Background Processor against database A.
3	This statement specifies that database B queries and tasks from class 8 are also processed by this Background Processor.

Example 2

```
a       CONTROL NAME M, QTIME 5, MAXQRY 5
b           DATABASE NAME A  CLASS 9, 12
```

Line no.	Explanation of statement
a	This statement specifies the input parameters for Background Processor M. The queue is checked for queries and tasks a minimum of every 5 minutes. The maximum number of queries in a batch is 5.
b	This statement specifies that database A queries and tasks from classes 9 through 12 are processed by this Background Processor.

Example 3

```
a       CONTROL NAME N, QTIME 2, MAXQRY 1
b           DATABASE NAME C  CLASS 14
c       BS00 LOG STAT
c       BT00 LOG STAT
c       BE00 LOG STAT
c       CE00 LOG STAT
c       DB00 LOG STAT
c       LG00 LOG STAT
c       QS00 LOG STAT WRAPNO 40
c       QT00 LOG STAT
c       QE00 LOG STAT
d       IM00 LOG
```

Line no.	Explanation of statement
a	This statement specifies the input parameters for Background Processor N. The queue is checked for queries and tasks a minimum of every 2 minutes. The maximum number of queries and tasks in a batch is one. In this case, queries and tasks are not batched.
b	This statement specifies that C queries and tasks from class 14 are processed by this Background Processor.
c	These statements instruct the Background Processor to write all nine log records to both the sequential log file and the communication file status log. WRAPNO 40 indicates that 40 query started records (QS00) are on the communication file before wrapping occurs (that is, query 41 will replace query 1, and so on).
d	This statement instructs the Background Processor to write the IM00 log record to the sequential log file.

Example 4

```
a       CONTROL NAME P PRINT
```

Line no.	Explanation of statement
a	This statement specifies that Background Processor P can only print reports (VISION:Bridge and Batch Simulator only).

Commands for Controlling Resources

VISION:Inform provides Foreground Processor commands for task handling and query control. By means of the SYSTEM user ID, the system administrator can use these commands on any query or task in the system. This section contains a brief description of each of the individual commands available for resource control. For detailed descriptions of the system administration commands, see Chapter 5, "System Administration".

ENABLE/DISABLE Commands

The DISABLE command changes the status of a currently AWAITING query or task to DISABLED status (for example, to defer processing).

The ENABLE command changes the status of a currently DISABLED query or task to AWAITING status.

A query or task has a current status of DISABLED if the Background Processor originally processing it encounters a query-related problem or if the query or task was specifically disabled. The query stays DISABLED until the ENABLE command changes the status of the query, or the query is purged.

An example of a query-related problem is when the background library does not contain all the field definitions used in the query because it was updated and the foreground library was not. You can successfully submit the query, because the definitions are present in the foreground library. However, the query fails because the two libraries are not synchronized.

You can display the current status of queries and tasks with the QSTATUS command.

These commands are not documented in the user documentation. It is the system administrator's option to inform individual users of their existence.

MAINT Command

The MAINT command displays platforms enabled, library information, and communication file information.

- This command also displays System Modifications (SMs) applied to the Background Processor named in the command, or, if you do not supply a name, the Foreground Processor.
- If you issue a MAINT command with no operand in Batch Simulator input, the SMs applied to the Batch Simulator display.

This command is not documented in the user documentation. It is the system administrator's option to inform individual users of its existence.

PCHECK Command

The PCHECK command displays information from the status log on the communication file for one or more Background Processors. Informational and diagnostic messages can both be displayed.

This command is not included in the user documentation. It is the system administrator's option to inform individual users of its existence.

PSTATUS Command

The PSTATUS command displays the status of one or more Background Processors.

PURGE Command

The PURGE command deletes a query or task in any status from the communication file. You can delete:

- All queries and tasks.
- All query and task output.
- An individual query or task.
- Output from an individual query or task, or selected queries and tasks.

If you (as the system administrator) purge a query or task, it is your responsibility to notify the user who submitted it.

QCHECK Command

The QCHECK command displays query information from the status log on the communication file for one or more Background Processors. Both informational and diagnostic messages display.

This command is not included in the user documentation. It is the system administrator's option to inform individual users of its existence.

QSTATUS Command

The QSTATUS command displays the status of submitted queries, output destination, number of output pages, processing class, and the databases being processed. You can limit the display to a specific status (for example, AWAITING), query, or data view.

- The SYSTEM user ID can display all users or limit the display to specific users.
- Other user IDs can display only their own queries.

You can request the display of a long version of QSTATUS (QSTATUS LONG) that includes the Background Processor name, the submit date and time, and the completion date and time of the queries.

TERM Command

After you determine which Background Processors are active using the PSTATUS command, you can use the TERM command to terminate a Background Processor.

Techniques for Controlling Resources

This section describes techniques to control user access to databases and optimize system resources. There are two types of techniques:

- Those that define data.
- Those that restrict access to data, impose processing options, and establish criteria for task or query execution.

The techniques described in this section conserve memory usage, shorten processing time, and limit data transmission.

Controlling Access Through File Definitions

You control access to data by creating custom file definitions that define a limited view of the database. Most databases contain many fields. In most cases, users only need access to a some of the fields in a database. You can define only those fields users need to access.

Define databases with single access paths, whenever possible. They are more efficient to process and the output they generate is less confusing to the user.

Limiting Access Through Profiles

You can also limit user access to data through the user profile. VISION:Inform requires every user to have a profile. The profile controls access to databases, segments, fields, and records.

You impose these controls through the specification of the optional profile commands EXCLUDE, RESTRICT, and SELECT.

- The EXCLUDE command denies users access to entire databases.
- The RESTRICT command restricts access to segments and fields of a database.
- SELECT restricts access to individual records based upon the criteria specified in the IF expression. The selection criteria apply to the root level of the database.

Restricting access to data through the user profile can be a more practical technique than creating customized file definitions. By using the profile technique, you can give users the definition for an entire database and their profile determines database items they can or cannot access.

For detailed information about the specification of user profiles, see Chapter 3, "VISION:Inform System Security".

Defining and Displaying Field Descriptions

Field descriptions help users determine the fields appropriate for their applications. Avoiding the selection of incorrect fields reduces use of system resources.

Defining Field Descriptions

As the system administrator, you define the field descriptions when you create the file definition using the Definition Processor. When you include the database name in the user profile, you make the field descriptions and data available to the user.

These descriptions can describe the content of fields and any special considerations for their use. Once defined, these descriptions are stored along with a copy of the database definition in the foreground and background libraries.

Displaying Field Descriptions

Users can display field descriptions by any of the following methods:

- By selecting the Main Menu Option 4 (Quick Query) or Option 5 (Quick Query Immediate Processing), then selecting a field from the Quick Query panel.
- By accessing these descriptions from a workstation and storing them on a disk to be used by the workstation client platform.
- By entering the GLOSSARY command for a definition.
- By displaying the FIELD DETAIL panel during a VISION:Bridge Full Screen Editor session.

For detailed information about the specification of user profiles, see Chapter 3, "VISION:Inform System Security". For detailed information about creating field descriptions, see the VISION:Inform *Definition Processor Reference Guide*.

Using Table Lookup Codes (Not Applicable to Immediate Response)

Brief codes can be stored in database fields that can cause a table lookup to occur. The table result field can provide an alternate description for the field.

For example, the value 01 can be stored in a database field containing the month. A reference to this field causes a table lookup and the result field produces the month of JANUARY.

The use of this table lookup technique can lead to significant savings in the size of databases.

For more information about tables, see *VISION:Inform Concepts Guide* and *VISION:Inform Definition Processor Reference Guide*.

Using Memory Optimized Processing

You can define IMS databases to be processed using memory optimized processing. Memory optimized processing limits the amount of memory required to process a database. The savings occur as a result of accessing and storing in memory only one occurrence of each database segment at a time.

You need to carefully consider the structure of the database before deciding to use memory optimized processing. This type of processing saves memory usage at the possible expense of excessive I/O, because of repeated retrievals of the segments due to looping. For this reason, it might not be suitable for processing all databases.

For more information about the use of memory optimized processing, see the *VISION:Inform Concepts Guide* and *VISION:Inform Definition Processor Reference Guide*.

Specifying Output Controls

The MAXPAGE and MCRPAGE parameters specify output control:

- The value you assign to the MAXPAGE parameter in PARMBLK controls how query (VISION:Bridge and Batch Simulator) report output is handled. It does not affect the subfile output.
- The MCRPAGE parameter controls how workstation client output is handled.

These parameters provide a means for controlling the amount of output stored in the communication file. When the amount of data equals or exceeds the MAXPAGE or MCRPAGE value, the Background Processor routes the output to the INFREPT data set of the Background Processor.

Depending on the LEVRPT specification in PARMBLK, report data can be deleted from the communication file. Task output is not deleted from the communication file when the MCRPAGE limit is exceeded.

For detailed information about these parameters, see the *VISION:Inform Installation Guide* for your environment.

Distributing Report Output

VISION:Inform provides a number of commands for handling the output from processed queries. The SYSTEM user ID can use these commands on any queries in the system while other users can only act upon queries submitted by their user ID.

The output from a processed query is handled in different ways depending on the specifications in the user's profile and the specifications included in the Submit panel for that query. The various options are described in the following sections and summarized in Figure 4-2.

Report Handling

When you submit a query for execution, VISION:Inform assigns a 4-digit **query number** to it. You use this number to route, view, disable, enable, and purge queries submitted through the communication file. You can also reference the query by its name.

A QSTATUS command displays both the query number and query name. The query number is particularly useful when there are several queries in the system with the same name.

Using the PRINT Parameter

Report output from processed queries returns to the communication file unless you specify on the Submit panel or SUBMIT command that output is to be routed to the printer. When you specify the PRINT parameter, the report output transfers to the INFREPT data set when the Background Processor that processed them terminates.

Not Using the PRINT Parameter

Report output from queries you submit without a PRINT parameter return to the communication file. You can view the output as often as you want, and you can route the output to another logical terminal or to the printer using the ROUTE command.

Note that reports are automatically purged from the communication file after printing unless you specify NOPURGE in the ROUTE command.

Submit Panel and SUBMIT Command

Reports are formatted according to the characteristics of the submitting terminal unless a logical terminal is specified in the parameter on the Submit panel or in the SUBMIT command. Routing reports subsequent to submission does not change the formatting characteristics.

Note: *Reports using alternate output formats are not bound by terminal characteristics.*

For a discussion of the ROUTE, PURGE, and VIEW commands, see the *VISION:Inform User Guide*.

Submit Information	Profile (Print Specified?)	QSTATUS (Does Query Show?)	Output Returned to	View at Terminal?	Route to Printer
SUBMIT PRINT CLASS	No	Awaiting/ Active Only	Printer†	No	Automatic
SUBMIT CLASS	Yes	Awaiting/ Active Only	Printer†	No	Automatic
SUBMIT CLASS	No	Yes	Communication File	Yes‡	Yes‡

† Output is sent directly to the INFREPT data set associated with the Background Processor that is processing the query. The report prints when the Background Processor that processed the query shuts down.

‡ The user can use all the report handling commands (ROUTE, VIEW, and PURGE).

Figure 4-2 Disposition of Processed Queries

Alternate Report Output Formats

In addition to the standard report output and subfiles, VISION:Inform provides four additional formats that can be used for the content of a report. A format in this context refers to how the data content for the report is prepared for output. When these alternate report output formats are specified, the data that would typically be prepared for a report page layout is instead prepared in one of the four special formats that are more suitable for use by other software tools. The output for these report formats is always directed to a file that is dynamically allocated by the background processor (or batch simulator), based upon the user ID of the submitter of the query, the report format, and entries in a special data set (DDNAME HCSCFG). Depending upon the format used, certain specifications

intended specifically for a printed page layout are syntactically accepted but functionally ignored. This is done to make it easy to change the report statement specifications from a form suitable for a printed page to an alternate output format and vice versa. VISION:Inform supports the following four alternate report output formats. The table below lists the specifications for the output format value used on a FORMAT or SUBMIT statement. See the *VISION:Inform User Guide* for further information.

Alternate Report Format	FORMAT/ SUBMIT Specification	Description
Comma-Delimited Data (.csv)	COMMA	Output data fields as a comma-delimited file
Tab-Delimited Data (.txt)	TAB	Output data fields as a tab-delimited file
Plain Text File (.txt)	PLAIN	Output report as a plain text file
HTML Document (.html)	HTML	Output report as an HTML document

Each of the formats is described in more detail in the next four sections.

Comma-Delimited Data (CSV)

The Comma-Delimited Data format (sometimes known as Comma-Separated Variables) is used for preparing data for import into spreadsheets or other tools that will accept this form of input. It formats the columns of each detail or summary line as a series of fields delimited by commas. This output is restricted to detail only (no summaries) or summary only (no detail) data. Column heading lines are also formatted as delimited data lines, unless column headings were suppressed. No page title lines appear on the output. (If you specify page title lines, they are ignored.) If the data for any column contains a comma, the column contents are bracketed by quotation marks (“). If the data for any column contains a quotation mark (“), the quotation mark is changed to an apostrophe (‘).

Tab-Delimited Data

The Tab-Delimited Data format is used for preparing data for import into spreadsheets or other tools that will accept this form of input. It is nearly identical to the Comma-Delimited Data format except that a tab character rather than a comma delimits the columns of data. If the data for any column contains a tab, the character representing the tab will be changed to a blank. No quotation marks, to bracket the column contents, are ever inserted.

Plain Text File

The Plain Text format is used for preparing data for insertion into other text documents. It prepares the report content without machine or platform-specific spacing control characters preceding each line. Blank lines are embedded into the output to simulate the line spacing that would normally appear on a printed report. A plain text report does not contain any page breaks. The page title and column headings appear once at the beginning of the report and only at NEWPAGE control breaks on a GROUP field.

HTML Document

The HTML format is used for preparing data for viewing by many users via a corporate intranet or worldwide via the Internet. It prepares the report content for viewing with a web browser. The HTML document uses the frames feature of modern browsers to render the report data suitable for viewing and scrolling. This method requires a template containing HTML frame specifications in the M4HTBASE library (partitioned data set). A default template is delivered with VISION:Inform in the INFORM.SRCLIB installation data set. You can customize the default template, which is known as a "style". The HTML format places the report output into a dynamically allocated partitioned data set. Members of this data set contain the HTML specifications and data for the report document. The member names in this data set are always $MAIN, HEAD, BODY, or LTOC. The LTOC member is present when any level 1 subtitle specifications are present in the report definition; otherwise, it is absent. These members must be transferred, using a suitable file transfer tool, to a server or local workstation where the web browser can access them. The browser session used to view the VISION:Inform HTML document must initiate the viewing via the $MAIN file (member).

Required Setup for All Alternate Report Formats

All of the alternate report formats require some initial setup so that the dynamically allocated report output files are successfully created. The two areas of concern for the setup are:

- Security authorization for the dynamically allocated report output files
- Overriding defaults for dynamic allocation parameters

Security Authorization

To define security authority (for example, CA-ACF2, CA-Top Secret, or RACF) for query output data sets, you need to understand the naming convention of the query output data sets that are created on the host when a query is run. All data set names are constructed with either four or five levels of qualifiers. The qualifiers and their descriptions are shown below.

Prefix	Optional entry dependent on the DATASETPREFIX parameter in the HCSCFG configuration file. If there is no DATASETPREFIX, then the generated DSN will have only four levels of qualifiers, starting with the user ID (see next entry). If DATASETPREFIX is used, this qualifier must be authorized for data set creation and deletion.
User ID	This is the VISION:Inform profile user ID. It is the second level DSN qualifier if DATASETPREFIX is used, or the high level qualifier if DATASETPREFIX is not used. If DATASETPREFIX is not used, it is this qualifier that must be authorized for data set creation and deletion.
Query Name	The name of the submitted query that created the alternate output format data set is used as the next level of qualifier for the DSN.
Time Stamp	An encoded date and time stamp is used for the next DSN level. This stamp will ensure uniqueness of DSNs.
DSN Type	The lowest level of the DSN is a constant based upon the output format type, as follows: HTML document - HTML Tab Delimited - TAB Comma Delimited - CSV Plain Text - TXT

To define security authority for the alternate output format report files:

1. Decide whether you want all alternate output format files created by VISION:Inform aggregated under one high-level qualifier, or under each user's user ID.
2. If you want everyone to share one high-level qualifier, specify the prefix in the HCSCFG data set of the background processor or batch simulator job, using the DATASETPREFIX parameter. If you do not specify the DATASETPREFIX, then the data set is cataloged with a prefix of the same name as the user ID.
3. Give each user ID a minimum security authority to read and delete data sets that begin with the authorized prefix.
4. You can define the security authority down to the user ID level, ensuring that the user who runs a query is the only user who can access the alternate output for that query. Defining the user IDs this way ensures that the query outputs, which are OS/390 data sets, are governed by the security policies of your system. The query outputs can be retrieved only by those users who have been given the authority to read the data sets.

Dynamic Allocation Parameters

The parameters used by VISION:Inform in the HCSCFG configuration file are described in the following table.

Parameter	Description
DATASETPREFIX=	The prefix value to be used as the high-level qualifier of the query output data set names. Enter any valid OS/390 data set qualifier, up to eight characters. This is an optional parameter. If this parameter is not specified, the user ID is used as the prefix.
PRIMARYSPACE=	The number of records for the primary space allocation of the query output data sets. The default is 10,000.
SECONDARYSPACE=	The number of records for the secondary space allocation of the query output data sets. The default is 10,000.
STORCLASS=	The storage class used when creating the query output data sets that are managed by the Storage Management Subsystem (SMS). This is an optional parameter that defaults to a value determined by the installation. It is ignored if SMS is not installed or is not active.
DATACLASS=	The data class used when creating the query output data sets that are managed by SMS. This is an optional parameter that defaults to a value determined by the installation. It is ignored if SMS is not installed or is not active.
MGMTCLASS=	The management class used when creating the query output data sets that are managed by SMS. This parameter is ignored if SMS is not installed or is not active.
UNIT=	The unit on which the query output data sets are allocated. It can be a specific device, or a certain type or group of devices. This is an optional parameter that defaults to a value determined by the installation.
VOLSER=	The specific volume serial number on which the query output data sets are allocated. This is an optional parameter that contains no default. If the parameter is omitted, a volume serial number is not used when allocating the query output data sets.

The first statement in the HCSCFG file must be:

```
<VISION:INFORM>
```

Code the parameters in the above table one per statement line. Your parameter can start in any column. Comments in the HCSCFG file are indicated by placing a semi-colon (;) in column 1 of the statement. A sample HCSCFG file is supplied in the INFORM.SRCLIB installation data set, member HCSCNFIG. This sample member is listed below:

```
<VISION:INFORM>
PRIMARYSPACE = 10000
SECONDARYSPACE = 10000
UNIT = SYSDA
; DATASETPREFIX =
; STORCLASS =
; DATACLASS =
; MGMTCLASS =
; VOLSER =
```

Required Setup for HTML Reports

You must perform some initial setup steps in order for HTML reports to be prepared properly. HTML reports can be created from any query processed by a background processor job, or by a query submitted in class 15 with the batch simulator. The setup steps are:

- Prepare the HTML templates for use by a background processor or batch simulator.
- Specify, in the background processor or batch simulator job, a JCL statement defining the location of the templates to be used for HTML reports.

Prepare HTML Templates

When a query specifies the alternate report output format as an HTML document, VISION:Inform uses templates containing HTML code to define the layout and appearance of the report content, when it is displayed as a document by a web browser. Default templates are delivered with VISION:Inform to use in generating HTML documents. These templates include HTML style specifications such as background color, text color, font, alignment, etc., that the browser uses when displaying the content of the document. A template representing a specific appearance is known as a style.

You can customize the default style delivered with VISION:Inform, as you prefer, although the default templates will work as delivered. At execution time, the HTML templates for the style must be present in a library. This library is identified with the M4HTBASE ddname. The default templates for VISION:Inform are in the INFORM.SRCLIB installation data set.

The template for each HTML style consists of 5 related members in the M4HTBASE library. Each template member name for a style must not be changed, although multiple versions of an M4HTBASE library can be maintained to allow

different background processors to use different styles. The following table describes the purpose of each of the 5 members defining a specific style. Note that the HTPRIX00 and HTPRIY00 members serve the same basic function, and either one or the other is used for each report. The HTLTOC00 member is only used when the HTPRIX00 member is chosen for a report.

M4HTBASE Library Member Name	Contains the HTML code that...
HTPRIX00	Defines the frames used for an HTML document that includes Level 1 Subtitle specifications.
HTPRIY00	Defines the frames used for an HTML document that does not include any Level 1 Subtitle specifications.
HTBODY00	Describes the Body frame for the document. This frame contains the report data in a layout similar to a printed page. The body frame contains all subtitle lines, detail lines, and summary lines specified for the report content.
HTHEAD00	Describes the Heading frame for the document. This frame contains the Page Title and Column Heading lines for the report, similar to a printed page.
HTLTOC00	Describes the Left-Hand-Table-of-Contents frame for the document. This frame contains the Level 1 Subtitle data for the report and can be used as a hyperlink index to the corresponding section of detail and/or summary data lines related to the subtitle data value.

Within the HTBODY00, HTHEAD00, and HTLTOC00 members is a section of HTML code delimited by the <STYLE> and </STYLE> tags. It is this section of HTML code that can be modified to create additional styles. To create a new style, copy the members from the default or existing user style and modify the HTML code within the style section, to provide the appropriate appearance of the new style. You must not modify, insert, or remove any HTML code outside of the style section of these members.

Detail and summary lines in the Body frame as well as Page Title and Column Heading lines in the Heading frame are inserted into the appropriate frames as pre-formatted text bracketed by the <PRE> and </PRE> tags. Subtitle lines in the Body frame are bracketed by the <H5> and </H5> tags. Level 1 Subtitle data in the Left-Hand-Table-of-Contents frame are bracketed by the <A> and </A> tags. The style section for each frame contains style specifications for each tag. Changing the style parameters for a tag changes the appearance of the related report data when it displays using the browser.

The Heading frame for the default FRAMESET definition (in members HTPRIX00 and HTPRIY00) contains the specification of SCROLLING=NO. To enable scrolling for this frame, in the HTPRIX00 and HTPRIY00 members change the specification to SCROLLING=YES. If you are familiar with HTML coding, you can modify or extend the FRAMESET and FRAME specifications in these two members to meet specific needs or preferences. The portion of the screen space reserved for each FRAMESET is a parameter that you can change to provide styles that can better meet the needs for a particular report. You can also define an additional frame to include a corporate logo or other static information to be displayed with each report.

When a VISION:Inform report is prepared as an HTML document, the report content is written out as either 3 or 4 members into the partitioned dataset that is dynamically allocated for the report. The following table shows the correspondence of the style members in the M4HTBASE library to the output member in the output dataset. Note that the $MAIN output member is copied from either the HTPRIX00 or HTPRIY00 template member. The LTOC output member is present only when the HTPRIX00 template member is used.

M4HTBASE Library Member Name	Action	Output Member Name
HTPRIX00	Copied as is when Level 1 Subtitle specifications are present.	$MAIN
HTPRIY00	Copied as is when Level 1 Subtitle specifications are not present.	$MAIN
HTHEAD00	Page Title and Column Heading lines are merged into this member at the “…..” placeholder location.	HEAD
HTBODY00	Subtitle, detail, and summary lines are merged into this member at the “…..” placeholder location.	BDYnnnnn
HTLTOC00	Level 1 subtitle fields are merged into this member at the “…..” placeholder location.	LTOC

The following HTML code represents each of the default style members provided with VISION:Inform.

Member HTPRIX00

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
```

```
<TITLE>Advantage VISION:Inform Report Type X</TITLE>
</HEAD>
<FRAMESET COLS="19%,*">
  <FRAME NAME="ltoc" SRC="ltoc.html" MARGINHEIGHT=0 MARGINWIDTH=0>
  <FRAMESET ROWS="13%,*" BORDER=1>
    <FRAME NAME="head" SRC="head.html" MARGINHEIGHT=2 MARGINWIDTH=2
           SCROLLING=NO>
    <FRAME NAME="body" SRC="body.html" MARGINHEIGHT=2 MARGINWIDTH=2>
  </FRAMESET>
</FRAMESET>
</HTML>
```

Member HTPRIY00

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Advantage VISION:Inform Report Type Y</TITLE>
</HEAD>
<FRAMESET ROWS="13%,*" BORDER=1>
  <FRAME NAME="head" SRC="head.html" MARGINHEIGHT=2 MARGINWIDTH=2
         SCROLLING=NO>
  <FRAME NAME="body" SRC="body.html" MARGINHEIGHT=2 MARGINWIDTH=2>
</FRAMESET>
</HTML>
```

Member HTHEAD00

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Advantage VISION:Inform Report Heading Frame</TITLE>
<STYLE>
<!--
BODY {color:black; background:aqua}
PRE  {color:black; background:aqua; font-size:x-small}
-->
</STYLE>
</HEAD>
<BODY>
<PRE>
.....     <Place Holder - Do Not Remove or Reposition>
</PRE>
</BODY>
</HTML>
```

Member HTBODY00

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Advantage VISION:Inform Report Detail Frame</TITLE>
<STYLE>
<!--
BODY {color:blue; background:white}
H5   {color:black; background:lightgrey; font-size:x-small;
      font-weight:bold; font-family:courier;
      text-align:left; line-height:normal}
PRE  {color:blue; background:white; font-size:x-small}
-->
</STYLE>
</HEAD>
<BODY><PRE>
```

```
.....     <Place Holder - Do Not Remove or Reposition>
</BODY>
</HTML>
```

Member HTLTOC00

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<BASEFONT SIZE=2>
<HEAD>
<TITLE>Advantage VISION:Inform Report Table of Contents Frame</TITLE>
<STYLE>
<!--
BODY {color:black; background:skyblue}
H4   {color:yellow; background:blue; font-size:medium; font-weight:bold}
A    {color:maroon; background:skyblue; font-size:x-small}
-->
</STYLE>
</HEAD>
<BODY>
<H4>Report Contents</H4>
<UL>
.....     <Place Holder - Do Not Remove or Reposition>
</UL>
</BODY>
</HTML>
```

Specify HTML Template Location

The JCL library that comes with the VISION:Inform installation (INFORM.JCL) contains the procedures for executing background processor and batch simulator jobs. These jobs are in the form of in-stream JCL procedures. One of the parameters in each procedure is the **HTMLTPL** parameter. You provide the name of the style library containing the templates described above in this parameter. The default is the installation source library (INFORM.SRCLIB) where the default templates are located.

Locating Alternate Output Format Report Files

When a VISION:Inform user submits a query that specifies one of the alternate output formats for a report, the report output for that query is not directed to the communications file as standard queries are, but is instead stored in a dynamically allocated OS/390 data set. Summary data for the alternate output format query is written to the communications file, and when the user issues the VISION:Inform VIEW command for the query (in READY status), they will see information similar to that shown below:

```
**OUTPUT FOR USER ATKGE02**
**ID SAMPLE ALTERNATE FMT RPT  **
**QUERY 1234-QUERY1  ,  CLASS 10**
**REPORT NO.  1, PAGE COUNT        **
**AGAINST DATABASE FINANCE  **
**PROCESSED 08/29/02    AT 17:52:41**
HTM='ATKGE02.QUERY1.H2941104.HTML', SIZE=     28K, RECS=    32
```

The first six lines are the standard VISION:Inform report header that is part of all report output. The seventh line shows the generated data set name for the alternate output format report file, along with a record count and an approximate file size in bytes.

Understanding PCB Selection

This section describes the PCB selection techniques used by Background Processor and by Immediate Response:

- A Background Processor searches a PSB for the appropriate PCB to use in processing DL/I databases. The way it selects a PCB depends upon several factors.
- Immediate Response uses a different technique than the Background Processor. The Immediate Response processor searches PSBs differently under CICS than under IMS.

Note that for IMS systems only, the PCBs for the system files must always be positioned in the PSB before those for the user databases.

Background Processor PCB Selection

The Background Processor searches the PSB and selects the PCB with a matching DBD name and the most restrictive processing option that fits. For instance, a PCB with PROCOPT=G is selected instead of one with PROCOPT=A.

The Background Processor tests the leftmost processing option of the PROCOPT parameter. The only options checked for are G or A. For example, PROCOPT=GD satisfies selection, but PROCOPT=DG does not because G is not the leftmost processing option.

The Background Processor issues a MK4D501 diagnostic message when the Background Processor does not find an acceptable PCB for the following reasons:

- The file definition specifies the name of a non-existent DBD.
- The available PCBs do not contain the correct processing option.

In addition to these conditions, unpredictable results occur if a PCB is used that specifies ascending sequential access (PROCOPT=GS).

Background Processor PCB Considerations

The following considerations pertain to PCBs used with the Background Processor:

- Primary or synchronized files of a logical data view must specify a PCB processing option of either G or A.
- One PCB must be supplied for each DL/I database defined.
- Multiple uses of the same database cannot share the same PCB.
- The PCB for DBFILE0 is selected first, then DBFILE1 through DBFILE9.

Immediate Response PCB Selection

Immediate Response searches for PCBs from the top of the PSB until it finds the first PCB that matches the DBD name. Unlike the Background Processor, Immediate Response does not check the PROCOPT parameter. However, there are exceptions to this general selection technique; these exceptions are described in the paragraphs that follow.

Under CICS

A default PSB name is provided in PARMBLK. You can modify this PSB name with your own PSB name. You can also specify PSB names in the user profile. Using this technique, you can specify a PSB for each database the user intends to access.

Immediate Response uses the PSBNAME on the INCLUDE statement to process the database requested in the query. It will use the default PSB in PARMBLK when the user profile does not contain a PSB.

Under IMS

You must specify a PSB that contains all the PCBs that Immediate Response will require online. There is only one PSB that contains all the possible PCBs. The PSBNAME in the user profile and the default PSB name in PARMBLK are ignored.

With Logical Data Views (LDVs)

This PSB contains all the PCBs for the DL/I databases included in the LDV. It schedules the PSB for each particular database specified in the INCLUDE statements.

Immediate Response issues the following diagnostic messages when it does not find an acceptable PCB:

- OG01 — a PCB was not found.
- OG02 — an LDV needed multiple PCBs for the same database and they were not found.

Understanding Database Calls

This section briefly describes VISION:Inform DL/I calls.

The types of calls vary by the type of processing being performed by VISION:Inform. Global optimization, at the root level only, causes VISION:Inform to perform a random read on the primary file to position into the database. From there on, VISION:Inform processes the database sequentially. All reads for synchronized files are random.

The types of processing are:

- Standard Processing.
- Multiple-Path Memory Optimized Processing.
- Single Path Memory Optimized Processing.

Standard Processing

With standard processing, all occurrences of all segments of the database are read into memory for each root segment. All query references to a database cause processing of the in-memory record.

Mode	Function	Segment	Boolean Expression
Sequential	GN	Root	N/A
Random	GU	Root	(Key=X) Where X = qualification statement.

Standard processing uses unqualified GNP calls for lower level segments.

Multiple-Path Memory Optimized Processing

VISION:Inform performs multi-path memory optimized processing when it is specified for a database in the logical data view and the PCB for that database has segments for more than one leg of the database. Since only one occurrence of each segment type is held in memory at any given time, each query reference to the database causes a read.

Mode	Function	Segment	Boolean Expression
Sequential	GN	Root	N/A
Random	GU	Root	(Key=X) Where X = qualification statement.

If automatic processing on lower level segments is in effect, the following calls are used:

- GN*F, then GN*–(KEY>LASTKEY).
- GU*–(KEY>LAST KEY) to reposition for the preceding occurrence of the other path.
- All positioning is by *V on higher level segments.

Single Path Memory Optimized Processing

Single path memory optimized processing is similar to multi-path memory optimized processing, except that the PCB contains segments for only one leg of the database.

Mode	Function	Segment	Boolean Expression
Sequential	GN	Root	N/A
Random	GU	Root	(Key=X) Where X = qualification statement.

Single path memory optimized processing uses the same root calls as multiple path memory optimized processing.

If automatic processing on lower level segments is in effect, the following calls are used:

- GN*F, then GN*–.
- All positioning is by *V on every higher level segment.

Using Subfiles (VISION:Bridge and Batch Simulator Only)

Subfile facilities enable users to output and save subsets of databases for future use. Using the EXTRACT command, the user specifies the name of the subfile to be output, the fields to be output, any selection criteria that apply, and whether or not VISION:Inform generates file definition statements for the subfile.

VISION:Inform automatically:

- Checks the user's profile to be sure it has the authority to create a subfile and if so, ensures that no restricted information is output.
- Outputs all or part of a database record, as specified in the query, to the file specified in the Background Processor JCL.
- Automatically generates file definition statements if requested in the query.
- Provides for up to nine subfiles to be output from a single batch of queries.

Preliminary Operations

For VISION:Inform to generate subfiles, you need to perform the following preliminary operations:

- Allocate space for each of the subfiles that could be output. Make this allocation prior to run time or at run time.
- Modify the JCL for the Background Processor to include a DD statement for each potential subfile.
- Verify that the user profile contains a SUBFILES parameter (provided on the PROFILE statement) identifying the ddnames of the subfiles that the user can generate. If SUBFILES is not specified in a profile, subfile generation facilities are not available to the user of that profile.
- Add the subfile ddname with an INCLUDE statement to the profiles of any users who will be interrogating the subfile after the file definition has been promoted to the background and foreground libraries.

The remainder of this section addresses each of these responsibilities.

Specifying Subfile Data Sets

Pre-allocate a sufficient amount of disk space for each of the subfiles that could be generated.

Assigning Subfile Ddnames

Before doing this, devise a scheme for assigning ddnames to the various subfiles. The ddnames are the subfile names that you use with the SUBFILES parameter in user profiles. Make each subfile name up to eight (non-blank) characters long (the first must be alphabetic, the rest alphanumeric).

If two or more users share a common subfile name in their profiles, one of them could inadvertently write over a subfile that has already been generated by another user. To avoid this, exclusively assign each subfile name to a particular user profile or specify the disposition of MOD so that the subfile can accumulate data from multiple users. In addition to naming conventions, note other operational considerations in the section Considering Subfile Operational Impact.

Subfiles can be fixed or variable length. The data set can only accept subfile data in the allocated format.

DD Statement Conventions

The following conventions apply to DD statements for subfile data sets (allocation or Background Processor executions):

- Match the subfile names listed in a user profile with the ddnames in the Background Processor JCL.
- Specify DSNAME, DISP, VOLUME, and UNIT parameters according to your installation standards.
- Specify the SPACE parameter as the number of tracks or cylinders only (as for a physical sequential data set).
- For the DCB parameters, you can:
  - Allocate fixed-length subfiles with RECFM=F or RECFM=FB.
  - Allocate variable-length subfiles with RECFM=VB.
  - Specify the record length and block size in LRECL and BLKSIZE; once determined, the generated subfile cannot exceed those values.
- If you do not specify RECFM, LRECL, or BLKSIZE, VISION:Inform supplies the following values the first time a subfile is generated to that data set (these values will not change for future subfiles output to that data set):
  - For a fixed-length subfile:

    RECFM=F
    LRECL=total length of generated fields
    BLKSIZE=total length of generated fields (same as LRECL)
  - For a variable-length subfile:

    RECFM=VB
    LRECL=total length of generated fields + 4
    BLKSIZE=same as the block size for report files

    The M4PARAMS option for report size is also the default buffer size for subfiles, hence, specify sufficient space to handle any subfiles you could create in the future.

See M4PARAMS in the *VISION:Inform Installation Guide* for your environment.

- No other parameters are necessary on the DD statements for allocating subfile data sets.

Specifying File Definitions for the Subfiles

You have the option to write your own file definitions for the subfiles. However, VISION:Inform will perform this function for you if the DEFINE keyword is specified on the EXTRACT statement.

Note: *If you are not going to output automatic file definitions, you need not include specifications for M4SUBF0 in your JCL for the Background Processor*.

If you choose not to write your own subfile file definitions, you must allocate a special data set (M4SUBF0) exclusively for VISION:Inform to write the file definition statements of generated subfiles. File definitions are automatically generated and written to the M4SUBF0 data set whenever the user specifies DEFINE as a parameter of the EXTRACT command in a query.

Every time a user specifies DEFINE in the EXTRACT statement of a query, VISION:Inform writes an RC statement to M4SUBF0 in addition to the file definition statements. M4SUBF0 is input to the Definition Convert Utility which puts the definitions in the definition library and removes surplus RC statements. Once in the definition library, use the Promote Process Utility to promote the definitions to the foreground and background libraries. (For information on these utilities, see the *VISION:Inform Utilities Guide* for your environment.)

The subfile specifications discussed in the section Specifying Subfile Data Sets apply to the definition data set with some exceptions:

- You must use M4SUBF0 as the ddname of the definition data set. When VISION:Inform outputs file definitions, it looks for M4SUBF0.
- The physical file name can be whatever you like.

VISION:Inform generates source code for the subfile file definition statements in the form of 80-byte records. Specify RECFM=F and LRECL=80 when you allocate the M4SUBF0 data set.

Subfiles must be promoted to the foreground and background libraries and specified on an INCLUDE statement in a profile before they can be used for queries. For information on defining files, see the *VISION:Inform Definition Processor Reference Guide*.

Specifying Subfile Generation

For VISION:Inform to generate subfiles and write them to the allocated data sets, you must include a DD statement in the Background Processor JCL for each subfile. To generate subfile definitions, you must include a DD statement for M4SUBF0.

Note the following when including JCL for subfiles:

- With DISP=OLD, there is no protection against accidentally writing over desirable data if the same subfile name is specified in subsequent queries. The same is true if two EXTRACTs are done to the same subfile or physical file in the same query. Data from the second EXTRACT will be written over the data from the first one.
- For the definition data set, M4SUBF0, specify DISP=MOD. This will accumulate all the definition statements of the generated subfiles, and prevent the former ones from being written over with new definition statements. After you promote the definitions written to M4SUBF0, you can clear the file and begin accumulating new subfile definitions. It is the system administrator's responsibility to see that the definition source statements are properly maintained and promoted to the foreground and background libraries.
- For all previously allocated subfiles except the definition subfile, specify DISP=OLD to allow new copies of a subfile to be placed in the same physical location on disk (writing over any old copy). Using DISP=MOD will concatenate the new data set to the end of an existing data set.

Example:

```
//M4SUBF0   DD   DSN=INF.SUBDEF,DISP=MOD
//SUBF010   DD   DSN=INF.SUB1,DISP=OLD
```

In the above example, definition statements will be written to M4SUBF0.

Specifying Profile Requirements for Subfiles

Users cannot generate subfiles unless the SUBFILES parameter is present in their profile. The specified subfile names belong exclusively to the user profile in which they appear and cannot be inherited by subordinate users.

The following is an example of a profile which enables the user to output three subfiles:

```
PROFILE PASSWORD YYY SUBFILES SUBF010,
   SUBF020, SUBF030
INCLUDE DATABASE PLANT
END INCLUDE
END PROFILE
```

If you have allocated space for subfiles SUBF010, SUBF020, and SUBF030, and added DD statements for these subfiles in the Background Processor JCL, this user can generate three subfiles using the EXTRACT command. For the proper format and use of this command, see the *VISION:Inform User Guide*.

If the user needs to access any of these three subfiles, or any other subfile later, you must add an INCLUDE command to the user's profile after you promote the definitions to the background and foreground libraries.

The VISION:Inform security system denies access to any databases not explicitly identified in the profile with an INCLUDE command or inherited from a group profile. As VISION:Inform does not allow you to include databases that do not exist in the profile, you must define a subfile in the background and foreground libraries before you specify it in an INCLUDE command.

While only one person could have the explicit ability to create subfiles, many people can have explicitly defined access to those subfiles. In these situations, it is important to communicate to users that by creating a subfile, they run the risk of writing over existing subfiles that could be of importance to someone else.

Considering Subfile Operational Impact

The following are considerations for using subfiles:

- A single query or batch of queries cannot generate more than nine subfiles, not including the definition subfile M4SUBF0.
- The Background Processor automatically batches queries in such a way that they do not generate more than nine subfiles in one pass of the database.
- When a subfile with variable-length fields is being generated, all fixed-length fields must appear before any variable-length fields. You will need to inform the users of any fields that are variable in length.
- It is your responsibility to inform users that there is a risk of writing over existing subfiles that could belong to someone else authorized to access that subfile.
- When you use the automatic file definition feature, the first field listed on the EXTRACT statement is designated as the key field for the subfile in the file definition statements. If this field is not the key field in the database, you might need to modify the file definition source.
- If an existing subfile is regenerated with new data in the same format, a new definition is not necessary.
- M4SUBF0 should be input to the Definition Convert Utility, which puts the definitions in the definition library.
- If you intend to accumulate definition statements of multiple subfiles in M4SUBF0, use DISP=MOD in the DD statement of M4SUBF0 in the JCL for the Background Processor.

Calling External Routines

The VISION:Inform call facility invokes external routines from queries.

- Using the CALL command, queries pass parameters to external routines which use the parameters and return results to the query. You can use the results returned from the call like any other temporary or database fields.
- External routines written in Assembler, COBOL, FORTRAN, PL/I, or virtually any language, are first assembled and link edited into their respective load or link libraries.
- By concatenating this link library with the program specified in the STEPLIB or JOBLIB of the Background Processor JCL, the external routine will then be available to queries through the CALL command.
- Users need only specify the name of the external routine followed by a list of positional parameters. These input and output parameters can be fixed-length database fields, temporary fields created with SET or LET commands, character constants, or numeric constants.

VISION:Inform takes a conventional approach to calls. That is, the called routine, not VISION:Inform, defines the parameter list requirements. Routines invoked by the CALL statement could modify only those values passed as parameters and not values or data structures in the VISION:Inform internal environment.

Implementing User Written External Routines

Users can write an external user routine in any programming language (Assembler, COBOL, FORTRAN, PL/I, and so on) as long as the appropriate programming conventions are observed.

- Link edit the program, as a reusable load module and place it in the VISION:Product STEPLIB or JOBLIB concatenation.
- To ignore operating system interrupts, link the Assembler interface (Assembler Interface Routine for COBOL) with the user subroutine. Specify the Assembler routine as the entry point of the generated module.

  This load module interface is the name that is called from the VISION:Bridge query.

  It is the interface that calls the link edited subroutine.

The following sections contain sample interface routines for COBOL, FORTRAN, and PL/I programs.

COBOL/Assembler Interface Module

The following figure shows an Assembler interface for calling a COBOL program from a query. If this interface is not used, the non-Assembler subroutines can abend when called by the Background Processor, because the COBOL program is not flagged as a subroutine.

Note: *Replace XXXXXXXX with the module ILBOSTP0 for an MVS® COBOL subroutine.*

```
ASMUSER   CSECT
          USING       *,15
          STM         0,15,SAVE        -- (save registers)
          CLI         SWITCH,C'Y'
          BNE         CALLUSER
          MVI         SWITCH,C'N'
          L           15,=V(XXXXXXXX)  -- (call COBOL interface)
          DROP        15
          BALR        14,15
          USING       *,14
          LM          0,15,SAVE        -- (restore registers)
          DROP        14
          USING       ASMUSER,15
CALLUSER  L           15,=V(USERPGM)   -- (call user routine USERPGM)
          SR          5,5
          ICM         5,8,=X'3n'
          SPM         5
          BR          15
SAVE      DC          16F'0'
SWITCH    DC          C'Y'
          END         ASMUSER
```

Figure 4-3 Assembler Interface Routine for COBOL

Disabling Interrupts

If you want all interrupts disabled, leave the ICM statement out completely, since the SR 5,5 sets the Program Mask to all zeros.

In the hexadecimal value 3n, the 3 is the condition code which has no meaning in the PSW. The 3 was arbitrarily chosen and can be replaced with any hexadecimal number. The n represents the Program Mask of the PSW.

The contents of this last half-byte determines the handling of the following interrupts:

Bit 1: Fixed Point Overflow

Bit 2: Decimal Overflow

Bit 3: Exponent Overflow

Bit 4: Significance.

Bit values

A 0 in a bit disables the interrupt and a 1 enables the interrupt.

X'30' disables all interrupts.

X'31' enables Fixed Point Overflow interrupt and disables all others.

.

.

.

X'3F' enables all interrupts.

If this interface is not used, the COBOL program can abend when called by VISION:Inform if the above noted data exceptions occur.

FORTRAN/Assembler Interface Module

The following figure shows an Assembler interface for calling a FORTRAN program from a query. If this interface is not used, the non-Assembler subroutines can abend when called by the Background Processor, because the FORTRAN environment has not been established.

```
ASMUSER    CSECT
           USING        *,15
           STM          0,15,SAVE        -- (save registers)
           L            15,=V(IBCOM)     -- (call FORTRAN interface)
           DROP         15
           BALR         14,64(15)
           USING        *,14
           LM           0,15,SAVE        -- (restore registers)
           DROP         14
           USING        ASMUSER,15
 CALLUSER  L            15,=V(USERPGM)   -- (call user routine USERPGM)
           SR           5,5
           ICM          5,8,=X'30'
           SPM          5
           BR           15
 SAVE      DS           16F"0"
           END          ASMUSER
```

Figure 4-4 Assembler Interface Routine for FORTRAN

If this interface is not used, the non-assembler routines may ABEND when called by VISION:Inform. This is because the FORTRAN environment has not been established.

PL/I Interface

If your external routine is a PL/I routine:

- You establish the environment prior to execution.
- You clean up the environment after the PL/I routine completes execution.

If you repeatedly call a PL/I routine in a loop, establish the environment once before the first call and clean up the environment once after the routine returns for the last time. This can save CPU time and disk I/O that would be used to establish the environment. See the following figures for examples of how this is done.

The BITRAN routine handles three major calling conditions:

- First call of the PL/I routine.
- Clean up call to the PL/I routine.
- Other intermediate calls.

Figure 4-5 shows the Assembler routine, BITRTN.

```
*                          ******************
*                          ***** MODIFY *****
*                          ******************
BITRTN    CSECT               IN THIS MODULE NAME IN VISION:INFORM CALL;
*                            IT MUST BE UNIQUE FOR EACH PL/I SUBRTN.
          ENTRY         PLICALLLA
PLICALLA  DS            0H
          USING         *,15
          LTR           1,1
          BZ            EXIT
*-------------------------------------------------------------------*
*    THE CODE FROM HERE TO LABEL "INIT" IS EXECUTED ONLY ON THE     *
*    FIRST CALL TO BITRTN.  IT SAVES REGISTERS IN THE SAVE AREA     *
*    SUPPLIED BY VISION:INFORM, AND CHAINS IN TWO LOWER LEVEL SAVE- *
*    AREAS, S1 AND S2.  IT SUPPLIES THE S2 SAVEAREA IN CALLS TO     *
*    THE PL/I ROUTINE ALLOWING PL/I TO CHAIN BACK TO THE SAVE-      *
*    AREAS CONTAINING THE PL/I ENVIRONMENT.  THIS INCLUDES THE      *
*    CHAIN CONTAINING THE PL/I DSA FOR OUR CONTROL BLOCK.           *
*    THE PL/I INTERLANGUAGE ROUTINE CHAINS 3 SAVEAREAS ABOVE THE    *
*    S1 SAVEAREA INSTEAD OF PUTTING THEM ABOVE THE SAVEAREA         *
*    SUPPLIED BY THE SYSTEM TO VISION:INFORM.                       *
*-------------------------------------------------------------------*
          CLI           ENVIRON,X'FF'          BYPASS ONE TIME CHECK
          BZ            INIT
          MVI           ENVIRON,X'FF'          CLOSE GATE
          STM           14,12,12(13)           SAVE CALLER'S REGS
          ST            13,S1+4                SAVE CALLER'S REG 13
          ST            13,S2+72               SAVE FOR ENVIRON RESTORE
          LA            11,S1                  GET OUR SAVEAREA
          ST            11,8(13)               AND CHAIN BACK
          LA            14,RET                 SET ONE TIME BACK
INIT      DS            0H                     SKIP POINT
          LA            13,S2                  SAVEAREA FOR PLI SUBRTN
*                                              ******************
*                                              ***** MODIFY *****
*                                              ******************
          L             15,=V(BITSUB)             NAME OF PLI ROUTINE
          BR            15
EXIT      L             15,=V(CLEANUP)            ASSM. MODULE THAT CLEANS
*                                                 UP PLI ENVIRONMENT ON
*                                                 LAST CALL (NO PARMS ARE
*                                                 PASSED ON LAST CALL).
          BR            15
```

Figure 4-5 Assembler Routine BITRTN (Page 1 of 2)

```
RET       L           13,72(13)
          ST          15,16(13)               SAVE PLI RETURN CODE
          LM          14,12,12(13)
          BR          14
S1        DC          A(0),A(0),A(S2),15F'0'
S2        DC          A(0),A(S1),A(0),16F'0'
          DC          V(PLIENV)
ENVIRON   DC          XL1`00'
          END         BITRTN
```

Figure 4-5 Assembler Routine BITRTN (Page 2 of 2)

The first call to the PL/I routine is issued in the first loop of the LOOP request. The last call is issued in the EOF request by having no parameters. The last call's parameter list pointer is logically zero. All intermediate calls are just like the first call, except that the ENVIRON field in the BITRTN assembler routine is set to X'FF.'

To complete the interface there are the two PL/I routines, BITSUB and PLIENV, and the assembler routine CLEANUP.

- BITSUB, shown in Figure 4-5, is the PL/I routine that makes the bit manipulation possible. The most important things to notice about the BITSUB code is the way the parameters are received from the application request LOOP. Note the DCLs (declares) in BITSUB and the TF statements in the LOOP request.
- PLIENV is a dummy PL/I routine that is called when no parameters are passed. It is used to bring in the established PL/I environment so it can be cleaned up.
- The CLEANUP assembler routine does all the work to clean up the environment.

BITSUB PL/I Routine

```
BITSUB: PROC (I1,I2,I3,I4) OPTIONS (COBOL,NOMAP) REORDER;
/*
/*  PURPOSE:   TO CONVERT FIELD, PASSED FROM VISION:INFORM, TO A CHARACTER
/*     STRING OF THE FORMAT:   " `XXX...'B " WHERE X IS ZERO OR ONE
/*     (E.G., `01010'B).  VISION:INFORM CAN THEN PROCESS THE STRING WITH
/*      PARTIAL FIELDING TO SIMULATE BIT MANIPULATION.
/*
/*  NOTE:  BECAUSE OF THE WAY PLI PASSES PARAMETERS, THE PROGRAM
/*     ACCEPTS "DUMMY" NUMERIC PARAMETERS (I1,I2,I3,I4) THEN OVERLAYS
/*     THEM WITH BASED VARIABLES HAVING THE CORRECT DATA TYPE.
/*
  DCL  BITSTRING BIT(32) BASED (P1),  /*  BIT STRING FROM VISION:INFORM */
       OFFSET  FIXED DEC(15) BASED (P2),  /* OFFSET INTO BIT STRING AT
                                          WHICH TO START PROCESSING */
       NUMBITS FIXED DEC(15) BASED (P3),  /* NUM OF BITS TO PROCESS */
       RESULT CHAR(255)BASED(P4);   /* OUTPUT FIELD RETURNED TO A/S */
  DCL (P1,P2,P3,P4) POINTER,
      (I1,I2,I3,I4),
```

Figure 4-6 BITSUB PL/I Routine (Page 1 of 2)

```
      (SUBSTR,ADDR) BUITIN;
  P1 = ADDR(I1);
  P2 = ADDR(I2);
  P3 = ADDR(I3);
  P4 = ADDR(I4);
/*   CONVERT INPUT FIELD TO OUTPUT CHARACTER STRING  */
  SUBSTR(RESULT,1,NUMBITS+3) =
      ''''  || SUBSTR(BITSTRG,OFFSET,NUMBITS) || ''B';
END;
```

Figure 4-6 BITSUB PL/I Routine (Page 2 of 2)

The JCL for this interface is included in Figure 4-9. Refer to Figure 4-9 and follow the instructions in Figure 4-7 and Figure 4-8 to interface PL/I to the Background Processor.

PLIENV PL/I Routine

```
PLIENV: PROC OPTIONS(MAIN)  REORDER;
/*  SKELETON PLI MAIN MODULE USED TO BRING IN PLI ENVIRONMENT.   */
    RETURN;
END PLIENV;
```

Figure 4-7 PLIENV PL/I Routine

CLEANUP Assembler Routine

```
CLEANUP  CSECT
         USING *,ENTER
         STM   R0,ENTERN,SAVEENT          SAVE REGISTERS
         DROP  ENTER
*-------------------------------------------------------------------*
*  CLEANUP CLOSES PLI ENVIRON.  PLI ADDS SAVEAREAS IN ZYGO-LINGUAL
*     CONTROL LIST (ZCTL) TO SET ENVIRON.  PLI ADDS 3 SAVEAREAS ABOVE
*     CALLER SAVEAREA, BETWEEN CALLER'S CALLER AND CALLER'S CALLER'S
*     CALLER.  TOP CHAIN IS MODIFIED TO POINT TO PLI INIT ROUTINE TO
*     CLOSE ENVIRONMENT.  REG14 AND CHAIN DATA IS SAVED IN 4-BYTE
*     GHOST SAVEAREA IN ZCTL.
*  THE ROUTINE COPIES GHOST SAVEAREA, ZAPS ZCTL TO PASS CONTROL TO
*     LABEL-RETURN AFTER ENVIRON IS CLOSED, POSITIONS AT SAVEAREA BELOW
*     ADDED SAVEAREAS, AND RETURNS.  PLI ROUTINES CLOSE ENVIRON & PASS
*     CONTROL TO LABEL-RETURN, THAT REMOVES HOOKS RETURNING TO CALLER.
*-------------------------------------------------------------------
          BALR  BASE,R0              GET BASE
          USING *,BASE               GET ADDRESSIBILITY
          L     WORK1,=V(IBMBILC1)   GET ADDRESS OF CONTROL BLOCK
          LTR   WORK1,WORK1          DOES THIS MODULE EXIST?
          BZ    CALLRETN             NO MORE PROCESSING
          L     ZCTL,0(WORK1)        GET ADDRESS OF ZCTL
          LTR   ZCTL,ZCTL            IS THERE A SAVEAREA?
          BZ    CALLRETN             NO ENVIRONMENT OPEN, RETURN
          CLI   4(WORK1),X'00'       IS BLOCK OPEN?
          BE    CALLRETN             BRANCH IF BLOCK CLOSED.
          MVC   GHOST(16),128(ZCTL)  GET GHOST SAVEAREA
          LA    WORK2,RETURN         GET NEW ADDRESS FOR SYSTEM RETURN
          ST    WORK2,140(ZCTL)      SAVE NEW REGISTER 14
          L     SAVE,48(ZCTL)        POINT TO SAVEAREA
          LM    LINK,BASE,12(SAVE)   RESTORE REGISTERS
          BR    LINK
```

Figure 4-8 CLEANUP Assembler Routine (Page 1 of 2)

```
*--------------------------------------------------------------------
*   LABEL RETURN GETS CONTROL WHEN ENVIRONMENT IS CLOSED.  GHOST SAVE
*     AREA WAS SAVED BY PREVIOUS CODE AND OUR ADDRESS IS PATCHED IN
*     GHOST SAVEAREA.  WHEN WE GET CONTROL, REG 13 POINTS TO CALLER'S
*     CALLER SAVEAREA.  CODE RESTORES REG 14 TO SAVEAREA AND RETURNS.
*--------------------------------------------------------------------
RETURN   BALR  BASE,R0
         USING *,BASE
         L     LINK,GHOST14
         ST    LINK,12(SAVE)
CALLRETN BALR  BASE,R0
         USING *,BASE
         LM    R0,ENTER,SAVEENT
         BR    LINK
R0       EQU   0
ZCTL     EQU   9
WORK1    EQU   10
WORK2    EQU   11
BASE     EQU   12
SAVE     EQU   13
LINK     EQU   14
ENTER    EQU   15
SAVEENT  DS  18F
GHOST    DS   3F
GHOST14  DS    F
         END
```

Figure 4-8 CLEANUP Assembler Routine (Page 2 of 2)

JCL to Interface PL/I to VISION:Inform

```
//PLICOMPL PROC MEM=
//*********************************************************************
//*             THIS STEP WILL COMPILE THE PLI PROGRAM                ***
//*********************************************************************
//PLICOMPL EXEC PGM=IEL0AA,
//         PARM='AB,A,MAR(2,72,1),NOMAP,NSXT,OF,OPT(TIME),X'
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD   DSN=YOUR.SOURCE.LIB(&MEM),               <== CHANGE (1)
//         DISP=SHR
//SYSLIN   DD   DSN=&&OBJ(&MEM),
//         SPACE=(CYL,(1,1,10)),
//         UNIT=SYSDA,
//         DISP=(MOD,PASS)
//SYSPRINT DD   SYSOUT=*
//         PEND
//*
//ASM      PROC MEM=
//*********************************************************************
//*           THIS STEP WILL ASSEMBLE THE ASSEMBLER PROGRAM          ***
//*********************************************************************
//ASM      EXEC PGM=(IEV90 OR IFOX00)                    <== CHANGE (2)
//         PARM='XREF(SHORT)
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD   DSN=YOUR.SOURCE.LIB(&MEM),               <== CHANGE (3)
//         DISP=SHR
//SYSLIB   DD   DSN=SYS1.MACLIB,
//         DISP=SHR
//SYSPUNCH DD   DSN=&&OBJ(&MEM),
//         SPACE=(CYL,(1,1,10)),
//         UNIT=SYSDA,
//         DISP=(MOD,PASS)
//SYSPRINT DD   SYSOUT=*
//         PEND
//LKED     PROC
```

Figure 4-9 JCL to Interface PL/I to VISION:Inform (Page 1 of 2)

```
//*********************************************************************
//*            THIS STEP WILL LINK EDIT AN OBJECT MODULE              ***
//*********************************************************************
//LKED     EXEC PGM=IEWL,
//          PARM='NOXREF,LIST,LET,NCAL,REUS'
//SYSUT1   DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLIB   DD   DSN=SYS1.PLIBASE,
//          DISP=SHR
//          DD   DSN=SYS1.SIBMBASE,
//          DISP=SHR
//SYSLMOD  DD   DSN=&&LOADLIB,
//          SPACE=(CYL,(1,1,10)),
//          UNIT=SYSDA,
//          DISP=(MOD,PASS)
//OBJLIB   DD   DSN=&&OBJ,
//          DISP=(OLD,PASS)
//SYSPRINT DD   SYSOUT=*
//          PEND
//*********************************************************************
//*              COMPILE AND ASSEMBLE ALL MODULES                     ***
//*********************************************************************
//*
//BITSUB   EXEC PLICOMPL,MEM=BITSUB
//*
//PLIENV   EXEC PLICOMPL,MEM=PLIENV
//*
//BITRTN   EXEC ASM,MEM=BITRTN
//*
//CLEANUP  EXEC ASM,MEM=CLEANUP
//*
//*********************************************************************
//*                LINK EDIT PLIENV TO ESTAB AN ENTRY                 ***
//*********************************************************************
//*
//PLIENV   EXEC LKED,PARM='XREF,SIZE=(925760,32000),LIST,LET'
//LKED.SYSLIN   DD  *
 INCLUDE OBJLIB(PLIENV)
 ENTRY          PLICALLA
 NAME           PLIENV(R)
/*
//*********************************************************************
//*                      LINK EDIT ALL MODULES                        ***
//*********************************************************************
//*
//LKEDALL  EXEC LKED,PARM='XREF,SIZE=(925760,32000),LIST,LET'
//SYSLIN   DD   *
 INCLUDE OBJLIB(BITRTN)
 REPLACE        PLICALLA
 INCLUDE OBJLIB(BITSUB)
 INCLUDE OBJLIB(CLEANUP)
 INCLUDE SYSLMOD(PLIENV)
 ENTRY          PLICALLA
 NAME           BITRTN(R)
/*


NOTES:
    CHANGE (1) - Your library where the PLI programs exists.
    CHANGE (2) - Either Assembler G (IFOX00) or H (IEV90)
    CHANGE (3) - Your library where our Assembler programs exist.
```

Figure 4-9 JCL to Interface PL/I to VISION:Inform (Page 2 of 2)

Character strings must be passed to the PL/I subroutine using arithmetic parameters declared as FIXED.

For example, the following VISION:Bridge statements create two temporary fields (PARM1 and PARM2) to be used as input and output fields by the called routine. The query calls the Assembler routine which then calls the user program (USERPGM).

```
SET PARM1 TYPE PACKED LENGTH 4 DECIMALS 2 VALUE 150.75
SET PARM2 TYPE CHAR LENGTH 10 VALUE 'CHARDATA'
CALL ASMUSER USING PARM1 PARM2
```

The corresponding PL/I code required is:

```
USERPGM: PROC(NUM_PARM,CHAR_PARM) OPTIONS (MAIN);
DCL NUM_PARM FIXED DECIMAL(7,2)
      CHAR_PARM FIXED;
DCL P1 POINTER;
P1 = ADDR(CHAR_PARM);
DCL = CHARDATA CHAR (10) BASED (P1);
```

CALL Considerations

When you write programs to be used in calls, consider the following:

- Call time is determined by the position in the database of the segment containing the field being passed as a parameter. Every time an occurrence of that segment is retrieved from the database, the call is performed.
- The execution frequency of a CALL statement is associated with the REPORT command that immediately follows. When a REPORT command immediately follows a CALL and specifies the CALL parameters as items to be reported, the same occurrences of those parameters that are passed to the user routine will be reported.
- Looping of parameter fields from different paths in the database hierarchy is not suppressed. Figure 4-10 shows the implications of looping initiated by a CALL statement.

Suppose a database has three segment types A, B, and C as shown below.

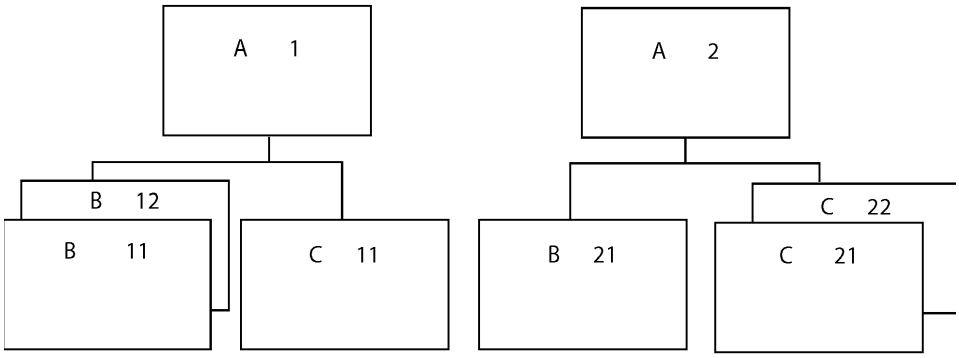


Figure 4-10 Sample Database to Show the Implications of Looping During a CALL

If user routine X is called and T1 and T2 are temporary fields used to return values from the call, the results will be as shown as follows:

Example Number	Command Statements	Parameter Values Passed to X Reported	Reported
1	CALL X USING A T1 T2 REPORT A T1	A1, T1, T2 A2, T1, T2	A1, T1 A2, T1
2	CALL X USING A T1 T2 REPORT B T1	A1, T1, T2 A1, T1, T2 A2, T1, T2	B11, T1 B12, T1 B21, T1
3	CALL X USING C T1 T2 REPORT A T1	C11, T1, T2 C21, T1, T2 C22, T1, T2	A1, T1 A2, T1 A2, T1
4	CALL X USING B C T2 REPORT B C,	B11, C11, T2 B12, C11, T2 B21, C21, T2 B21, C22, T2,	B11, C11 B12, C11 B21, C21 B21, C22

Figure 4-11 Interaction of the CALL and REPORT Statements

Notice that the first reference to a field in a lower level segment, be it the CALL statement or the REPORT statement, starts a loop on that segment. This is most clearly demonstrated in Examples 3 and 4.

In Example 3, the reference to field C in the CALL statement begins a loop on the segment containing field C. As a result, although only field A is reported, a value of A and T1 is returned for each occurrence of C. A2 is reported twice, once for the occurrence of C21 and once for C22.

Again, in Example 4, reference to field B in the CALL statement causes repetitive reporting of field C. When a REPORT statement immediately follows a CALL and specifies the CALL parameters as items to be reported, a loop has already been started on the first lower level segment referenced in the CALL statement.

Identifying Problems with External Routines

VISION:Bridge performs only a syntax check on the CALL parameters. Other errors, such as erroneous parameters or an erroneous subroutine can occur once VISION:Inform passes control to the user routine.

Erroneous Parameters:

- Parameter missing — the user passes fewer parameters than the called routine expects.
- Parameter passed with incorrect attributes — this can be due to parameters being passed in the wrong order for the data type or the length of a parameter not matching the corresponding length or type in the user routine.

Erroneous Subroutine:

Undesirable results can occur for the following reasons:

- There is a logical error in the user routine.
- Control is not returned to VISION:Inform.
- The user routine name does not exist in the proper program library.

These types of errors are the responsibility of the user.

Controlling Storage

VISION:Inform dynamically allocates working storage for the Foreground and Background Processors.

- If the Foreground Processor requires additional online storage, you must increase the region size to allow VISION:Inform to acquire more storage for the needed area.
- To obtain additional batch working storage space for the Background Processor, you must increase the region size.

Controlling Foreground Library and Communication File Space Utilization

When you run the VISION:Inform backup and restore utilities against the foreground library and the communication file, the utilities generate space utilization information that you can use to modify file sizes.

The backup and restore utilities write the space utilization information to the INFPRINT summary listing. For additional information on the utilities and INFPRINT, see the *VISION:Inform Utilities Guide* for your environment.

With the space utilization information, the system administrator can now determine:

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

Using VISION:Inform with DB2 (Not Applicable to Immediate Response)

This section presents the special requirements and considerations for using VISION:Inform to access DB2 databases. VISION:Inform can communicate with DB2 using any of three attach facilities provided with DB2:

- TSO Attach Facility
- IMS Attach Facility
- Call Attach Facility

The choice of which facility to use depends upon the requirements of your particular application and your installation standards.

TSO Attach Facility

You can use the TSO Attach Facility to access DB2 databases using the Background Processor. Any file type or database can be accessed concurrently within your query when using the TSO Attach Facility. Sample CLISTs provided with VISION:Inform in the INFORM.JCL library contain the proper sample statements for use with the TSO Attach Facility.

Make the system name and plan name in the DSN and RUN statements in the CLIST correspond to the system and plan names specified when the DB2 access module MARKSQLT was preprocessed, compiled, and link edited with the TSO Attach Facility interface module DSNELI. This process can be performed when VISION:Inform is installed at your installation.

For further information regarding the DSN and RUN command parameters, see the *IBM DB2 for OS/390® Application Programming and SQL Guide*.

IMS Attach Facility

You can use the IMS Attach Facility when the VISION:Inform Background Processor is executed in the BMP region. When determining the user ID to authorize use of DB2 tables, note that the Background Processor is considered a message processing program. Make the system and plan names used with BMP Background Processors correspond to the system and plan names specified when the DB2 access module MARKSQLI was preprocessed, compiled, and link-edited with the IMS Attach Facility.

For further information regarding DB2 applications in an IMS/VS environment, see the *IBM DB2 for OS/390 Application Programming and SQL Guide*.

Call Attach Facility

The Call Attach Facility is an alternate DB2 access facility for DB2 applications. You can use the Call Attach Facility with VISION:Inform whenever the DB2 parameter name is included with a logical data view or the DB2 Background Processor control statement is provided.

Make the application plan name specified on the DB2 parameter statement correspond to a plan name specified when the DB2 access module MARKSQLC was preprocessed, compiled, and link edited with the Call Attach Language Interface module DSNALI. You normally perform this process when you install VISION:Inform. No special JCL statements are required by the Call Attach Facility.

Logging Background Processor Activity

VISION:Inform makes available two optional log files that contain information about events as they occur during the execution of a Background Processor. The information stored on these files is determined by commands input through the INFIN data set when the Background Processor starts. A sample set of

commands, provided as part of the installation, logs all available information to both files. If none of the commands are used, no logging information is written to either file. Computer Associates recommends that you use these log files.

- The first file is a sequential log with the ddname INFLOG, which is available to external programs.
- The second is a status log maintained on the communication file. You can display the information on this log with the Foreground Processor commands described in the section Commands for Controlling Resources.

The files are independent and may or may not contain the same information, including data about Background Processor start up and termination, query start and completion, databases available, errors encountered during processing, and the logging control statements themselves. For the format of the logging control statements, see in the section Controlling Background Processors.

Sequential Log File

The sequential log file can contain the following record types:

BE00 — Background Processor error record.

BS00 — Background Processor startup record.

BT00 — Background Processor termination record.

CE00 — Control statement error record.

DB00 — Database record.

IM00 — Informational message record.

LG00 — Logging control record.

QE00 — Query error record.

QS00 — Query started record.

QT00 — Query completion record.

These record types are also the commands that cause their respective records to be written to the log files.

A file definition for the sequential log file is provided as part of the installation (member LOGFD in the supplied definition library), along with a sample query (member LOGQUERY in INFORM.SRCLIB) to report the data on the log file. All of the data on the file is in character display format and each record has a common header portion 34 bytes in length. The records are fixed length, 200 bytes long.

Each Background Processor writes its own log files while it executes and no provision is made within VISION:Inform to automatically preserve the data in a sequential log file across individual Background Processor executions.

A description of the header portion and the field names from the file definition LOGFD are as follows:

HEADER PORTION – 34 Bytes

Field Name	Length	Description
PROCNAME	8	Background Processor name.
MSGDATE	6	Date record was written (M4PARAMS TODAY format).
MSGTIME	6	Time record was written (HHMMSS).
RECTYPE	4	Record type, one of the ten listed above.
TEXTNUM	2	Used for record types BE00, BT00, CE00, and QE00. Contains '00' in the first record of one of these types. The second and successive records of the same type contain a sequential number starting at '01'. Each continued record has the same RECTYPE.
UNUSED	8	Reserved for future use.

The balance of each record type (RECTYPE) beyond the header portion is described below.

BS00

The run parameter information completes this record type, containing information from all the Background Processor control statements, except the DATABASE statement.

Field Name	Length	Description
MINQRY	3	MINQRY value for the Background Processor run.
MAXQRY	3	MAXQRY value for the Background Processor run.
QTIME	3	QTIME value from the CONTROL statement or blanks.
MAXTIME	4	MAXTIME value from the CONTROL statement or blanks.
MESSAGEQ	1	Not used.
PRINT	1	Contains a 'Y' if you specify PRINT on the CONTROL statement or an 'N' if not.

Field Name	Length	Description
DB2SYS	8	Contains the DB2 system name from the DB2 statement.
DB2PLAN	8	Contains the DB2 plan name from the DB2 statement.
FILL	12	Reserved for future use.
WRAPNO	8	Contains the WRAPNO value from the QS00 statement.

BE00, CE00, QE00, BT00, and IM00

These records have only message text following the header portion. TEXTNUM in the header portion specifies the continuation record number when multiple messages are present for the record type.

Field Name	Length	Description
TEXTDATA	133	Message text.

DB00

The database information fills out this record type with information from the DATABASE statement or the defaults if no statement is present.

Field Name	Length	Description
DATABASE	8	Database name.
FRMCLASS	2	The first or only value for the CLASS keyword.
TOCLASS	2	The second value for the CLASS keyword or blanks.
HELDSTAT	8	The value for the HELD keyword or blanks.

LG00

There is one of these records for each logging control command specified in the INFIN data set to the Background Processor. If all ten commands are used, there will be ten of these records.

Field Name	Length	Description
LOGTYPE	4	Record type from the control statement.
DEST1	8	Destination, LOG, or blank.
DEST2	8	Destination, STAT, or blank.

QS00 and QT00

These two record types contain information about query start and query completion.

Field Name	Length	Description
QRYNUM	4	Query number generated by the submit.
QRYNAME	8	Query name from the submit or blanks.
DBNAME	8	Database name from QUERY statement.
USERID	8	User ID submitting the query.
OWNERID	8	Same as USERID above unless the query belongs to different user ID.
SUBDATE	6	Date query was submitted (M4PARAMS TODAY format).
SUBTIME	6	Time query was submitted (HHMMSS).
CLASS	2	The submit class of the query.
ROUTING	8	Report destination (PRINT, terminal ID, or blanks).
MAXLINES	8	Contains value from Submit panel maximum number of lines or blanks.
MAXITEMS	8	Contains value from Submit panel maximum number of roots or blanks.
SOURCE	1	Contains a 'Y' if Submit panel specified generating source or 'N' if not.

Field Name	Length	Description
RPTPAGES	15	Contains the number of output pages generated by this query. This field is on record type QT00 only.
NUMREPTS	3	Number of reports for this query.
BATCHIND	1	'Y' or 'N' indicating whether the query was batched.

The data in these records provides valuable information. Use the sample query provided as part of the installation to report the data.

Status Log

Use the Foreground Processor to access the data in the status log file. The INFIN logging control commands for the Background Processor specify what data is recorded to the status log, which is maintained on the communication file.

Two VISION:Inform online commands, QCHECK and PCHECK, display the information in the status log. Use these commands to view the activity of the Background Processor while it is executing. For the format of these commands, see Chapter 5, "System Administration".

QCHECK and PCHECK are documented only in this book. The documentation can be made available to individual users at the discretion of the system administrator.

Each Background Processor writes its own status log while it executes and no provision is made within VISION:Inform to preserve the data in a status log across individual Background Processor executions. Procedures can be established external to VISION:Inform to preserve a sequential log file after the Background Processor has terminated.

Optimizing Database Access

The OPTMODE parameter in the PARMBLK determines the database optimization method used by VISION:Inform. (PARMBLK is discussed in the *VISION:Inform Installation Guide* for you environment.) There are three methods of optimization:

- Full optimization at all database levels (dynamic tuning).
- Optimization at the root level only (dynamic tuning).
- No optimization (static tuning).

Full Optimization

In an optimized search method, VISION:Inform completely bypasses the records not needed. This reduces processing time and more efficiently accesses the database.

VISION:Inform uses the full optimization method if OPTMODE=1 (the default value) in the PARMBLK and your query meets the following requirements:

- The query must contain a "global" conditional phrase. The conditions must pertain to all the statements in the query; the query cannot include IF and ELSE statements.
- The "global" conditional phrase must be in a SELECT IF, REPORT IF, or EXTRACT IF statement.
- The query can contain multiple SELECT statements and only one REPORT or EXTRACT statement.
- The SELECT, REPORT, and EXTRACT statement containing the "global" conditional phrase must not be preceded by LET or END statements.
- The conditional phrase must name the key fields or database search fields that are character (not numeric) for IMS databases.
  - Any field can be named for DB2 tables.
  - For VSAM and sequential files, only the root segment key (which must be character and less than 16 bytes) can be named.
  - Partial fielding of the named field is only allowed for DB2 tables and the relational operators can only be EQ or NE.
- The relational operators EQ, LT, LE, GT, and GE can be used in the conditional phrase; NE can only be used with DB2 tables and IMS databases.
- For IMS databases, either the logical connector AND or the logical connector OR can be used in the conditional phrase; combinations of both ANDs and ORs cannot be used together.
- Compound conditional phrases (a series ANDed or ORed together with a level of nesting) must have all of the simple conditional phrases within the compound optimizable. See the next section for the definition of optimizable conditional phrases.
- Compound conditional phrases are broken down into simple compound phrases. With IMS databases there can only be up to eight simple phrases in a query or batch of queries.
- If the database is HDAM, optimization is done on lower level segments only (not on the root).
- If any query in a batch does not have a "global" conditional phrase for a segment, that segment will not be optimized for that batch.

If your query does not meet these requirements or the batch of queries has conflicting requirements, VISION:Inform uses no optimization (OPTMODE=3).

Optimizable Conditional Phrases

The following five simple conditional phrases are the only types of phrases that are "optimizable" in the global conditional phrase. All other types of simple conditional phrases are considered non-optimizable.

In the following examples, RL represents a relational operator such as EQ or GE.

1. FIELD RL CONSTANT or CONSTANT RL FIELD
2. FIELD RL FIELD (where both fields are in the same segment)
3. FIELD RL SERIES (For example, FIELD = 1, 3, 7, 11)
4. FIELD RL RANGE (For example, FIELD = 10 to 50)
5. PF(FIELD) RL CONSTANT or CONSTANT RL PF(FIELD)

File Types	Allowable Conditional Phrases
DB2 tables	Examples 1 through 5
IMS Databases	Examples 1 through 4
VSAM and sequential files	Examples 1 and 4

Optimization at the Root Level

In an optimized search method, VISION:Inform completely bypasses the records not needed. This reduces processing time and accesses the database more efficiently.

VISION:Inform optimizes database access at the root level if OPTMODE=2 in the PARMBLK and your query meets the following requirements:

- The query must contain a "global" conditional phrase. The conditions must pertain to all the statements in the query. You cannot include an ELSE statement.
- "Global" conditional phrases must be in SELECT IF, REPORT IF, EXTRACT IF, or IF statements.
- The conditional phrase must name the key field of the root segment of the database. Partial fielding is not allowed.
- The conditional value must be a literal enclosed in quotation marks, regardless of the field type of the root segment key.
- The relational operators EQ, LT, LE, GT, and GE can be used in the conditional phrase; NE cannot be used.
- The queried database cannot be HDAM.

If your query does not meet these requirements, VISION:Inform uses no optimization (equivalent to OPTMODE=3).

Examples of Forcing Access Optimization

The following examples cause VISION:Inform to optimize database access.

Example 1.

```
QUERY DATABASE CUSTOMER
REPORT ORDERNO IF CUSTNO = '00115'
END REPORT
END QUERY
```

The optimized search strategy in this example contains a global conditional phrase in the REPORT statement immediately following the QUERY statement. Only information for customer 00115 will be accessed by the query. CUSTNO is the key field in the root segment and it is on the left side of the conditional phrase.

Example 2.

```
QUERY DATABASE CUSTOMER
IF CUSTNO = '00048' AND INSTNO = '23468'
REPORT ...
END REPORT
END QUERY
```

The optimized search strategy in this example contains a global conditional phrase in the IF statement immediately following the QUERY statement. All segments for customer 00048 will be accessed using the optimized search strategy.

The conditional phrase also contains the AND logical operator and a test for installation 23468. The segments that apply to this customer will be searched using the optimized strategy to locate the segments related to the 23468 installation.

Example 3.

The query in this example does not qualify for access optimization, because the conditional phrase does not appear immediately after the QUERY statement:

```
10 QUERY DATABASE CUSTOMER
20 LET QTYBKORD = QTYBKORD + 100
30 REPORT ITEMORD QTYBKORD IF CUSTNO = '00115'
...
...
...
```

It processes with no optimization.

No Optimization

When the OPTMODE parameter of PARMBLK is set to 3, VISION:Inform does not optimize database access. This is known as static tuning. In the case of static tuning, you must create your own WHERE clauses and SSAs as appropriate, because this method of optimization does not generate qualification of any kind. WHERE clauses could be included in the relational definition and SSA would be included using preselection requests cataloged with the database definition as an logical data view (LDV).

Optimization Versus Access Strategy

It is important to understand the difference between VISION:Inform optimization and the general access strategy used for any database.

Optimization

VISION:Inform optimization consists of generating predicates to be used in:

- WHERE clauses of SQL SELECT statements, or
- segment search arguments (SSAs) for DLI calls, or
- start/end search values for VSAM files,

thereby qualifying the access method READ operation for efficiency. VISION:Inform optimization does not control when an actual READ is performed — but does add qualification, if possible, to the READ when it does occur.

Access Strategy

Access strategy, which is independent of the access method, determines when a READ of a file occurs.

When determining the access strategy, VISION:Inform must take into account the possibility of multiple queries being batched together in a single pass of the database. VISION:Inform cannot process a database in such a fashion as to skip a READ of record for one query that may need to be made available to another query in the same batch.

You can affect when the READ occurs by the structure of a query.

Example 1.

The following example demonstrates how you can effect the access strategy. The example which assumes a DB2 logical data view with two DB2 tables joined together, will be used to show that if the query is structured with all qualification on the REPORT statement, it would appear as follows:

```
QUERY DB2LDV
REPORT TABLE1-FIELD TABLE2-FIELD IF TABLE1-FIELD = 'A' AND,
                                    TABLE2-FIELD = 'B'
END REPORT
END QUERY
```

In the above format, READs will always occur for TABLE1 and TABLE2 fields, prior to any comparisons being made on either TABLE1 or TABLE2.

If the query is re-structured in one of the ways as shown in the following examples, fewer READs will occur on TABLE2.

Example 2.

```
QUERY DB2LDV
SELECT TABLE1-SEGMENT IF TABLE1-FIELD = 'A'
SELECT TABLE2-SEGMENT IF TABLE2-FIELD = 'B'
REPORT TABLE1-FIELD TABLE2-FIELD
END REPORT
END QUERY
```

or

```
QUERY DB2LDV
SELECT TABLE1-SEGMENT IF TABLE1-FIELD = 'A'
REPORT TABLE1-FIELD TABLE2-FIELD IF TABLE2-FIELD = 'B'
END REPORT
END QUERY
```

In Example 2, READS will be done on the TABLE1 field, then the TABLE1 comparison is made. If, and only if, the TABLE1 comparison is TRUE, the TABLE2 field will be read and the TABLE2 comparison made.

Chapter

5

System Administration

This chapter describes the system administrator commands that you can use to examine and control the execution of queries, tasks, and Background Processors, as well as review the contents of the foreground library.

To use these commands, you must log on to VISION:Inform and select Option 1 (Operation Facilities) or Option 3 (Report Facilities) from the Main Menu.

Note: *See the Main Menu in* Figure 2-2.

After selecting Option 1 or Option 3, the system prompt, ?:, displays in the upper left corner of the screen. The prompt indicates that you are on the Command Input screen and can enter the commands discussed in this chapter.

This section contains an overview of the system administration commands, command syntax, and diagnostic messages. The following sections describe each of the system commands.

System administration commands examine or control the execution of queries, tasks, and Background Processors.

- Some commands are only available for use by the SYSTEM user ID.
- Selected system administration commands can be used by authorized users. It is at the discretion of the system administrator to provide these commands to authorized users.

DISABLE	ENABLE
MAINT	PCHECK
PSTATUS	PURGE
QCHECK	QSTATUS
TERM	

Note: *You can view glossary information in the full screen editor in the DataView, Fields, and Fields Detail panels.*

The following commands examine the foreground library or database definitions:

```
GLOSSARY

LISTLIB
```

The following command exits the Command Input screen and return to the Main Menu:

```
QUIT
```

Command Syntax

Begin each statement with a command.

- For some commands, there will not be any mandatory parameters following the command.
- For most commands, you follow the command with one or more parameters and associated operands (related names or values).

For example, when you display information on a database, you use the GLOSSARY command. In its complete form, you can use:

```
GLOSSARY DATABASE CUSTOMER SEGMENT ORDER
```

In this command, the parameters are DATABASE and SEGMENT. The DATABASE operand CUSTOMER identifies the database. The SEGMENT operand ORDER specifies that you want the ORDER segment, not the entire database.

Parameters

Enter parameters followed by their operands (related field names or values) in any order in a statement. Using the above example, you could also enter:

```
GLOSSARY SEGMENT ORDER DATABASE CUSTOMER
```

The parameters also have a system defined order (for this example, the first shown). When the parameters are entered in this order, they can be omitted. You enter:

```
GLOSSARY CUSTOMER ORDER
```

VISION:Inform identifies the elements according to their positions in the input sequence.

In cases where a parameter is followed by several field names or values, they are all assumed to belong to that parameter until another parameter keyword is encountered or the end of the statement is reached.

In many cases, system defaults are assigned to the parameters. In these cases, if you do not specify a value, the default value is used.

Separators

Separate the names, commands, parameters, and values that make up a statement from one another by one or more spaces. Do not embed spaces in commands, parameters, names, or values.

- You can use a comma in place of, or in addition to, spaces to improve readability.
- Parentheses and operators also serve as separators.
- Spaces are optional if parentheses or operators are present.

For example, when you remove queries or tasks from the communication file, you use the PURGE command. To purge three tasks (TASK2, TASK3, and TASK4), you can use any of the following:

```
PURGE TASK2 TASK3 TASK4
PURGE TASK2, TASK3, TASK4
PURGE, TASK2 TASK3 TASK4
```

All three statements have the same effect.

Continuation

Normally, you enter statements one per terminal line. If a statement does not fit on a single input line, you can continue it on subsequent lines by ending each line (except the last) with a comma. Do not split names, commands, constants, and operators from one line to the next.

For example, to purge a series of tasks (TASK1 through TASK9), use:

```
PURGE TASK1 TASK2 TASK3,
TASK4 TASK5 TASK6 TASK7,
TASK8 TASK9
```

Validating Commands

All Command Input statements are validated as you enter them at the terminal. Validation takes two forms:

- Ensuring that the statement is syntactically correct and field names are used properly.
- Ensuring that the statement is consistent with previously entered statements.

When you enter an incorrect statement, the system returns an appropriate diagnostic message and displays the Command Input prompt. You can then enter the corrected statement.

For a list of messages, see *VISION:Inform Messages and Codes*.

System Administration Commands

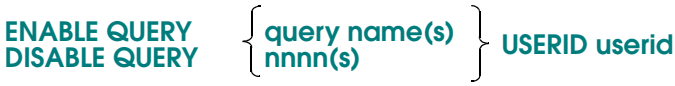
The system administration commands are described in the following sections. Each section describes a command, its syntax, and shows examples. Note that:

- Parameters enclosed in braces { } represent options from which you can select one or more.
- System commands and parameters appear in uppercase.
- Entries printed in lowercase are informational entries (such as field names) that you provide.
- Many parameters are optional if they are entered in the order specified. The description indicates if an entry or parameter is required.
- Default entries are indicated in the description.

ENABLE Command and DISABLE Command

ENABLE sets the specified queries to AWAITING status if they are currently disabled. DISABLE sets the specified queries to DISABLED status if they are currently awaiting.

Syntax:



Parameters:

Indicates the queries or tasks whose status is to be changed. Enter the name or 4-digit query number. You can enter multiple query names or query numbers.

USERID userid

Specifies the user who submitted the query or task. Can be used only by SYSTEM user ID. If this is omitted, the default is the current user.

The DISABLE and ENABLE commands are documented in this book only. It is the system administrator's option to make the commands available to specific users.

Command examples:

```
ENABLE QUERY XYZPRINT
ENABLE 1650
ENABLE FINREPT USERID FIN123
DISABLE 2491 2492
```

GLOSSARY Command

The GLOSSARY command displays the information in a database definition.

Syntax:

GLOSSARY DATABASE database name { **SEGMENT** / **FIELD** } { **segment name** / **field name** }

Parameters:

DATABASE database name

Required entry.

SEGMENT segment name

Displays all the fields in a segment.

FIELD field name

Displays the information on a specified field.

SEGMENT and FIELD are mutually exclusive. If neither is specified, the entire database glossary is listed.

Command examples:

```
GLOSSARY DATABASE FINANCE SEGMENT PROFCENT
GLOSSARY FINANCE FIELD ACTCHGMM
GLOSSARY FINANCE
GLOSSARY DATABASE FINANCE
```

Response example:

```
GLOSSARY CUSTOMER ORDER
CUSTOMER - IMS DATABASE FILE   DATE CREATED - 92.111   TIME 12.25.125
    NAME    TYPE   START  LENGTH   DEC  OUT-LEN  KEY  SEGNAME   PARENT
ORDCMPLT    CHAR    29      1              10         ORDER     CUSTOMER
        IS THE ORDER COMPLETE? (Y OR N)
ORDDUDAT    CHAR    22      6               8         ORDER     CUSTOMER
        DATE THE ORDER IS DUE
ORDERNO     CHAR     1      5               7         ORDER     CUSTOMER
        ORDER_NUMBER
        THE FIVE-DIGIT ORDER NUMBER
```

```
ORDINVGN    CHAR    28      1             10         ORDER     CUSTOMER
ORDPONUM    CHAR    17      5              8         ORDER     CUSTOMER
        PURCHASE_ORDER_NUMBER
ORDRDATE    CHAR     8      6              8         ORDER     CUSTOMER
        ORDER_DATE
ORPERSON    CHAR    14      3              8         ORDER     CUSTOMER
        ORDER_PERSON
        INITIALS OF PERSON WRITING THE ORDER
SEG20FIL    CHAR    30     10             10         ORDER     CUSTOMER
        UNUSED
***END OF DEFINITION***
```

Display Glossary Information From the Full Screen Editor

You can also obtain glossary information for a data view during a full screen editor session for queries by issuing the FIELDS primary command. The query must contain at least a valid QUERY statement and have been validated prior to using this command. The system responds with the following panel listing the fields for the current data view.

```
Editor2 ------- Computer Associates - Editor ------ Name: AQUERY   Type: QUERY
                                      +-------------------------------------+
000100 QUERY FINANCE                  | Fields2 Computer Associates - Fields|
000110 SELECT IF ACCOUNT_NUMBER GT '10 | Dataview: FINANCE      More:     +   |
000200 REPORT PROFIT_CENTER_ACCOUNT_NU | _ ACCOUNT                           |
000300 END REPORT                     | _ ACCOUNT_AMOUNT                    |
000400 END QUERY                      | _ ACCOUNT_BUDGET_AMOUNT             |
                                      | _ ACCOUNT_CHARGE_MONTH              |
                                      | _ ACCOUNT_CHARGE_PERIOD             |
                                      | _ ACCOUNT_CHARGE_YEAR               |
                                      | _ ACCOUNT_DATE                      |
                                      | _ ACCOUNT_DAY                       |
                                      | _ ACCOUNT_DESCRIPTION               |
                                      | _ ACCOUNT_MONTH                     |
                                      | _ ACCOUNT_NUMBER                    |
                                      | _ ACCOUNT_TYPE                      |
                                      | _ ACCOUNT_YEAR                      |
                                      | _ ACCTCNT                           |
                                      | _ DATECNT                           |
                                      |                                     |
                                      | Command ===>                        |
 Command ===>                         | F1 =Help          F7 =Backward      |
 F1 =Help     F2 =Save    F3 =Exit     | F8 =Forward       F12=Cancel        |
 F7 -Backward F8 =Forward F9 =Validate +-------------------------------------+
```

Figure 5-1 Editor Fields Panel

Obtain detailed information for a specific field by placing an 'S' next to the desired field and pressing ENTER. The system displays the Fields Detail panel as shown in Figure 5-2.

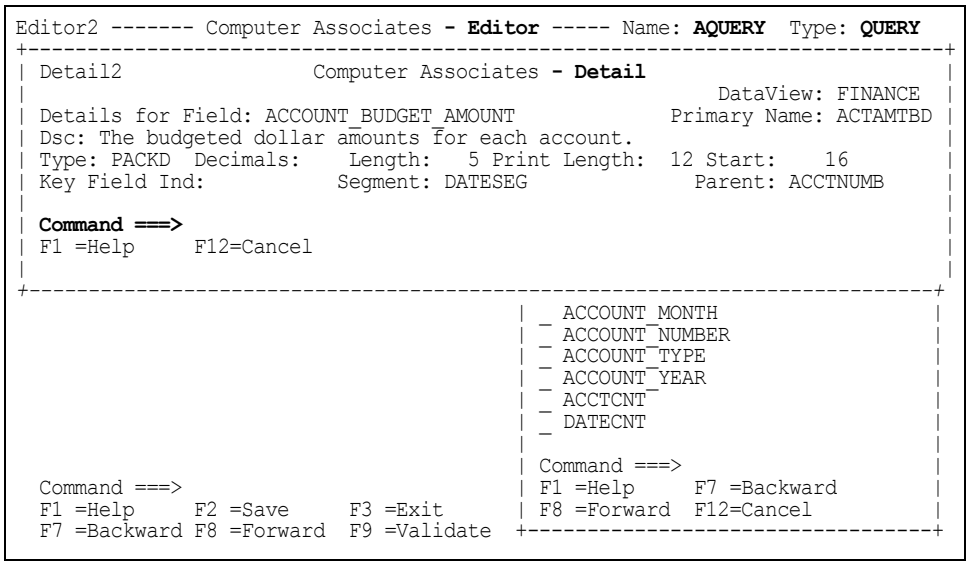


Figure 5-2 Fields Detail Panel

LISTLIB Command

The LISTLIB command lists items stored in the foreground library.

Syntax:

LISTLIB DATABASE

Parameter:

DATABASE

Lists all accessible databases.

Command examples:

Example 1.

LISTLIB DATABASE

Response example:

listlib database

DATAVIEW	TYPE	CREATED
BADFIN	NON-DLI	89166
CUSTITEM	LDV	89166

```
CUSTOMER        DLI         89166
FINANCE         NON-DLI     89166
ITEM            DLI         89166
```

Example 2.

You can also add the following optional operands to LISTLIB:

```
LISTLIB PROFILE
LISTLIB QUERY
```

The PROFILE operand, available only to the SYSTEM userid, lists the user profiles in the system. The QUERY operand lists items of type QUERY and STMTS.

Example 3.

You can enter the LISTLIB command without any operand, as follows:

```
LISTLIB
```

List Data Views From the Full Screen Editor

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

You can obtain a list of available data views during a full screen editor session for a query using the DATAVIEW command. The DATAVIEW primary command displays the DataView panel as shown in the following figure.

```
Editor2 ------- Computer Associates - Editor ------ Name: AAAA     Type: QUERY
                                    +-------------------------------------+
000100 QUERY FINANCE                | DataVw2 Computer Associates - DataView|
000110 LET PROFNO = 1               |                          More:     +  |
000200 REPORT PROFNO LILIAN         |   Name      Type  Date Promoted       |
000300 END QUERY                    | _ ASSETVW   LDV         2/24/97       |
                                    | _ BADFIN    OS          2/24/97       |
                                    | _ CUSTITMP  LDV         2/24/97       |
                                    | _ DBLOLYM   LDV         2/24/97       |
                                    | _ DB2TEST   DB2         2/24/97       |
                                    | _ FINANCE   OS          2/24/97       |
                                    | _ FINIVP    LDV         2/24/97       |
                                    | _ FINLDV    LDV         2/24/97       |
                                    | _ GOODFIN   OS          2/24/97       |
                                    | _ IN6OLYM   DLI         2/24/97       |
                                    | _ IN6PLNT   DLI         2/24/97       |
                                    | _ IN6TEST   DLI         2/24/97       |
                                    | _ IN6TES1   DLI         2/24/97       |
                                    | _ LOGFD     OS          2/24/97       |
                                    | _                                     |
                                    | Command ===>                          |
Command ===>                        | F1 =Help      F7 =Backward            |
F1 =Help     F2 =Save     F3 =Exit  | F8 =Forward  F12=Cancel               |
F7 =Backward F8 =Forward  F9 =Validate+-------------------------------------+
```

Figure 5-3 DataView Panel

Select a data view by placing an 'S' next to the particular data view and pressing ENTER. The Fields panel displays as shown in Figure 5-1. Use the LOCATE primary command in the DataView panel to position the display to a desired point in the data view list.

MAINT Command

The MAINT command displays the options installed and the System Modifications applied for the Foreground Processor or the specified Background Processor.

Syntax:

MAINT NAME processor name(s)

Parameter:

NAME processor name(s)

NAME is optional.

- If you include NAME, the system displays the maintenance status of the named Background Processor at the time it was last executed.
- If you omit NAME, the system displays the options installed and the System Modifications applied to the current Foreground Processor environment.

This command is documented in this book only. It is the system administrator's option to make it available to specific users.

Command examples:

```
MAINT
MAINT EXTRTDLI
```

PCHECK Command

The PCHECK command displays information from the status log on the communication file for the specified Background Processor. Specific logging control statements with TARGET STAT are required for the parameters to generate the displays.

Syntax:

PCHECK NAME processor name(s) { **STATUS** | **DATAVIEW** | **OPTIONS** | **ERRORS** }

Parameters:

NAME processor name(s)

Specify at least one Background Processor name.

STATUS

Default. Displays the run parameters and informational, error, and termination messages for the named Background Processors. Requires the BS00, BT00, and BE00 control statements.

DATAVIEW

When specified, displays the database and class information for the named Background Processors. The new status of previously held queries also displays. Requires the DB00 control statement.

OPTIONS

When specified, displays the logging control options for the named Background Processors. Requires the LG00 control statement.

ERRORS

When specified, displays any control statement diagnostic messages logged by the Background Processors to the status log. Requires the CE00 control statement.

This command is documented in this book only. It is the system administrator's option to make it available to specific users.

Command examples:

```
PCHECK NAME PROC10
PCHECK PROC25 OPTIONS
PCHECK PROC10 DATAVIEW
```

PSTATUS Command

The PSTATUS command lists the status of Background Processors that process data views (databases). You can list Background Processors that are currently active and available to process data. In addition, you can list Background Processors that have previously been used, but are not currently active.

Syntax:

PSTATUS NAME processor name(s) ALL

Parameters:

NAME processor name(s)

Indicates the names of the currently active Background Processors for which a status request is needed. The display shows the current status of each Background Processor listed — awaiting, processing, and so on.

ALL

Displays the status of all Background Processors that have been executed with the current communication file, both those that are currently active and those that have run previously, but are not currently active.

ALL and a list of names are mutually exclusive. If neither is specified, the status of all currently active Background Processors displays.

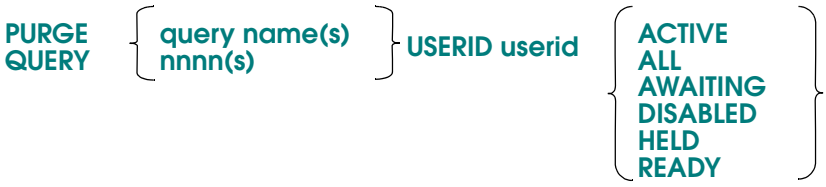
Command examples:

```
PSTATUS
PSTATUS NAME PAYPROC
PSTATUS ALL
```

PURGE Command

The PURGE command deletes one or more queries or tasks.

Syntax:



Conditions for specifying the PURGE command keywords: QUERY, ACTIVE, ALL, AWAITING, DISABLED, HELD, and READY:

- You must specify at least one of the following keywords: QUERY, AWAITING, ACTIVE, READY, DISABLED, HELD, ALL.
- You can use ACTIVE, ALL, AWAITING, DISABLED, HELD, and READY individually or in combination.

Parameters:

QUERY { query name(s) / nnnn(s) }

Indicates the queries or tasks to be purged. Enter the name or 4-digit number. You can enter multiple names or numbers. When you specify QUERY without the ALL operand and multiple queries exist with the same name, only the first query or task with that name will be acted upon.

USER userid

Specifies the user who submitted the query or task to be purged. This keyword is available to SYSTEM user ID only.

ALL

When specified with no other parameters, all queries and tasks belonging to that user are purged. When specified with AWAITING, ACTIVE, DISABLED, HELD, or READY, all queries and tasks AWAITING, ACTIVE, DISABLED, HELD, or READY are deleted.

When specified with QUERY, all queries and tasks with that QUERY name or names are purged.

AWAITING

Purges all queries and tasks awaiting processing.

ACTIVE

Purges all queries and tasks currently executing.

DISABLED

Purges all queries and tasks that have been DISABLED due to problems encountered when attempting to process the queries.

HELD

Purges all queries and tasks that have been HELD due to database unavailability.

READY

Purges all queries and tasks that have finished executing, including report output.

Command examples:

```
PURGE QUERY SHOTPUT ALL
PURGE ALL
PURGE QUERY 0145
PURGE SHOTPUT ALL READY
```

Response example:

```
purge fin1 fin2
 KF07** QUERY 6245 - FIN1      PURGED
 KF07** QUERY 6414 - FIN2      PURGED
 KF01** PURGE COMPLETED
```

QCHECK Command

The QCHECK command displays the query status log information and the query processing diagnostic messages for the specified Background Processors.

- Logging control statements QS00, QT00, and QE00 with TARGET STAT are required to generate the displays.
- The QCHECK command is a system administration command and an editor command.

Syntax:

Parameters:

NAME processor name(s)

Specify at least one Background Processor name. If no other operands are present, all query information displays.

DATAVIEW data view name

When specified, the display is limited to queries against the specified data view (database).

USERID userid

When specified, the display is limited to queries for the specified USERID. This keyword can only be used by the SYSTEM user ID.

When specified, the display is limited to the query specified by the query name or query number.

This command is documented in this book only. It is the system administrator's option to make it available to specific users.

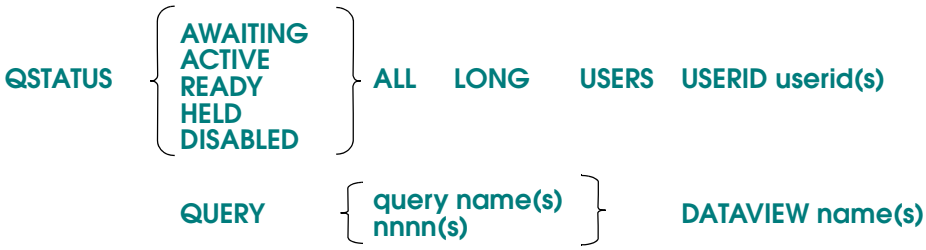
Command examples:

```
QCHECK NAME PROC10
QCHECK PROC25 FINANCE
QCHECK PROC10 DATAVIEW FINANCE USERID JUANITA
```

QSTATUS Command

The QSTATUS command gives the status of submitted queries.

Syntax:



Parameters:

AWAITING

Lists queries which have not been selected for processing by a Background Processor. Mutually exclusive with ALL.

ACTIVE

Lists queries which are currently being processed. Mutually exclusive with ALL.

READY

Lists queries which are completed and ready for viewing or delivery to the workstation. Mutually exclusive with ALL.

HELD

Lists queries which have been placed in “hold” status, because a data view required was unavailable.

DISABLED

Lists queries which have been disabled due to a query-related problem encountered during execution. Mutually exclusive with ALL.

ALL

Default. This option lists queries with all the above statuses and is mutually exclusive with the above keywords.

LONG

Causes a 2-line format of the display to be produced, which includes submit date and time, end date and time, and Background Processor name.

USERS

Lists queries for all user IDs. For use by user ID SYSTEM only. Use with care; this option is time consuming. Mutually exclusive with the USERID keyword.

USERID userid(s)

Lists queries for the specified user IDs only. For use by the SYSTEM user ID only. Mutually exclusive with the USERS keyword.

QUERY { query name(s) / nnnn(s) }

Lists the status of the specified queries only. Mutually exclusive with the DATAVIEW keyword.

DATAVIEW name(s)

Specifies the specific name or names of the data views for which queries are to be listed. Mutually exclusive with the QUERY keyword.

Command examples:

```
QSTATUS AWAITING
QSTATUS QUERY ABCD
QSTATUS DATAVIEW CUSTOMER
```

Response example:

```
QSTATUS  USERID  RON

QUERY #  -NAME      STATUS    DEST      PAGES     DATAVIEW  CLASS     USERID
4055     -RONCOMM   READY     A215       7        CUSTOMER  10        RON
4067     -GRANDSUM  READY     N012      12        CUSTOMER  10        RON
0411     -GRANDSU2  READY     N012      10        CUSTOMER  10        RON
```

QUIT Command

The QUIT command terminates the Command Input session and returns to the Main Menu.

Syntax:

QUIT

TERM Command

The TERM command terminates Background Processors.

Syntax:

TERM NAME processor name(s)

Parameter:

Name processor name(s)

A required entry. Specifies the name or names of the Background Processor (as specified in the Background Processor control statement) to be shut down. The processor will terminate after the batch completes.

You can have multiple entries. Names of active Background Processors are available with the PSTATUS statement.

Command examples:

```
TERM NAME BGPROC
TERM BGPROC ALLDBS
```

Response example:

```
term bgproc
 TMO1**BACKGROUND PROCESSOR BGPROC      WILL TERMINATE
```

The USERID Parameter

The SYSTEM user ID uses the USERID parameter to access queries and tasks belonging to all other users. The following commands can use the USERID parameter:

DISABLE	ENABLE	LIST
LISTLIB	PURGE	QCHECK
QSTATUS	ROUTE	RUN
SUBMIT	VIEW	

For example, to purge query MONTHLY saved with the user ID R2D2, enter:

```
PURGE QUERY MONTHLY USERID R2D2
```

The LIST and RUN commands can be used with the parameter USERID by all users, not just the SYSTEM user ID.

The user can only specify the user ID of a parent in the profile hierarchy, unless the logon user ID is SYSTEM.

It is the responsibility of the system administrator to furnish this information to authorized users as it is not documented elsewhere.

Chapter

6

Using the Batch Simulator

The VISION:Inform Batch Simulator provides the batch environment user with the same capabilities as those available to the online user.

- With the Batch Simulator, all of the system administration commands and VISION:Bridge language commands available in your online system can be processed in batch, thereby giving access to the VISION:Inform system even if the online monitor is unavailable.
- The difference between the Batch Simulator version and the online version is the interactive facilities, Quick Query and the Full Screen Editor, are not available for system administration and developing queries.

Batch Simulator JCL

The JCL required to run the Batch Simulator is delivered with your VISION:Inform system. (See the *VISION:Inform Installation Guide* for a sample.) Your Batch Simulator input statements are inserted in this set of JCL.

Batch Simulator Commands

There is no Logon panel or Main Menu in the Batch Simulator. Therefore, the first statement in Batch Simulator input statements must be a LOGON command (see the section Batch Simulator Commands).

Query, profile, and system administration commands are identical to those available in your online system, and they are composed as they would be online. Since the interactive facilities of Options 1, 3, and 7 on the Main Menu and the Full Screen Editor of Options 2 and 6 on the Main Menu are not available, you enter the input statements with a line editor.

This chapter describes how to specify input statements in the line editor, as well as the Batch Simulator.

Modes

The Batch Simulator version of VISION:Inform has different modes of operation. The commands you specify relate directly to a particular operation mode.

There are three modes:

Modes	Prompt	Used for
System mode	?:	System administration.
Input mode	n:	Entering new queries, profiles, or collections of statements.
Edit mode	EDIT:	Modifying and saving queries and profiles.

These modes are connected to one another through the various VISION:Inform commands. By entering the appropriate command, you switch to a different mode. This is called mode switching. Since your commands are processed in batch and not online, you must keep track of the mode.

Note: *The numbering for the Input mode (n:) prompt begins either with sequence number 10 (for the first line) or the sequence number following the last line in your edit work area.*

Figure 6-1 is a pictorial representation of the VISION:Inform modes.

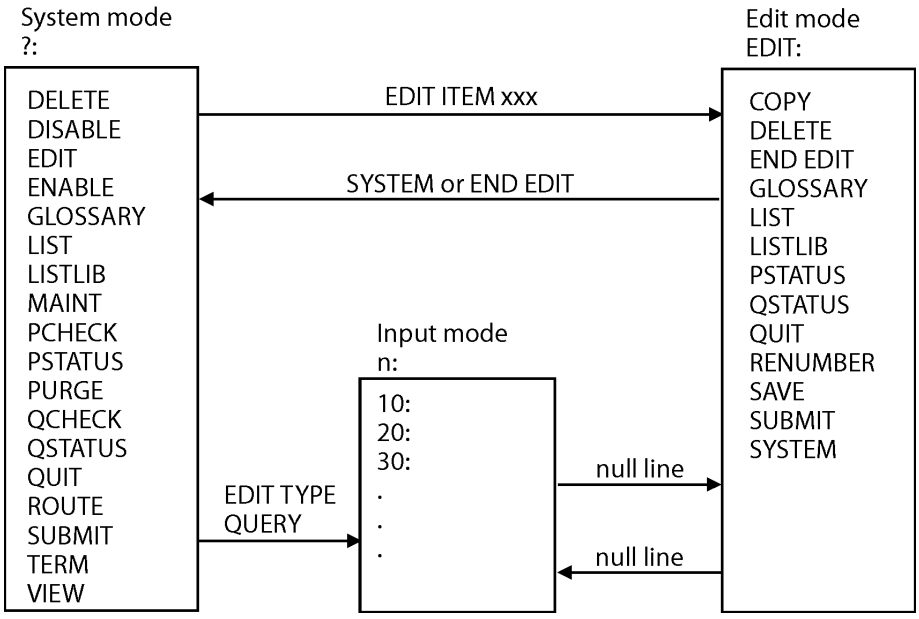


Figure 6-1 Batch Simulator Modes and Their Commands

System Mode

System mode is always the first VISION:Inform mode. From the system mode, you can:

- Control the operation of the Foreground Processor.
- Request information about queries, profiles, and file definitions from VISION:Inform.
- Enter any of the other modes.

The prompt for the system mode is ?:.

You can enter the following commands while in the system mode:

DELETE	DISABLE	EDIT
ENABLE	GLOSSARY	LIST
LISTLIB	MAINT	PCHECK
PSTATUS	PURGE	QCHECK
QSTATUS	QUIT	ROUTE
SUBMIT	TERM	VIEW

You can enter most of these commands from Options 1, 3, and 7 on the Main Menu.
For information on most of these commands, see **Chapter 5, "System Administration".**

For additional information on GLOSSARY, LISTLIB, PSTATUS, PURGE, ROUTE, SUBMIT, and VIEW commands, see the *VISION:Inform User Guide*.

The DELETE, EDIT, and SUBMIT commands are only used in Batch Simulator input statements and are discussed in the section Batch Simulator Commands.

Edit Mode

You enter edit mode (EDIT:) from the system mode (?:) by typing the following command:

EDIT item type

You can specify the parameter as TYPE QUERY, TYPE STMTS, or TYPE PROFILE. If you do not specify TYPE, the type defaults to QUERY.

In edit mode, you can:

- Make changes to any of the queries, profiles, or collections of statements that were created using the Batch Simulator and saved in the foreground library.
- Perform the information-gathering functions that you can perform in the system mode.
- Store and change queries and profiles.
- Submit queries for processing.

The prompt for the edit mode is EDIT:.

Line Editor Functions

As part of the line editor functions, you can insert, replace, or delete statements from queries and profiles. You can list query and profile statements, renumber them, or copy them.

From the edit mode (EDIT:), you can enter the input mode or return to the system mode (?:).

- To enter the input mode (n:), enter a blank line.
- To return to the system mode, enter:

  END EDIT

The following commands can be entered while in edit mode (EDIT:):

COPY	DELETE	END EDIT
GLOSSARY	insert/replace	LIST
LISTLIB	PSTATUS	QSTATUS
QUIT	RENUMBER	SAVE
SUBMIT	SYSTEM	

You can enter many of these commands from Options 1, 3, and 7 on the Main Menu.

For additional information on COPY, GLOSSARY, LIST, LISTLIB, PSTATUS, and QUIT commands, see the *VISION:Inform User Guide* for information on these commands.

The COPY, DELETE, END EDIT, insert/replace, RENUMBER, SAVE, SUBMIT, and SYSTEM commands are used in the Batch Simulator and are described in the section Batch Simulator Commands.

Input Mode

You enter input mode (n:) from the system or edit mode.

- If you are in system mode, you enter input mode (n:) by entering one of the following commands:

  EDIT TYPE QUERY

  or

  EDIT TYPE PROFILE

  or

  EDIT TYPE STMTS

  or

  EDIT

- If you are in the edit mode, you enter the input mode by entering a null line (that is, enter a blank line in your input statement).

Once you enter input mode:

- The Batch Simulator line editor prompts you with n:.
- The statements you enter are automatically numbered.
- The numbering either begins with sequence number 10 (for the first line) or the sequence number following the last line in your edit work area.

You leave the input mode by entering a null line (a blank input statement). When you leave input mode, you return to edit mode.

You can enter all VISION:Bridge query statements in input mode. See the *VISION:Inform User Guide* and *VISION:Inform Reference Summary* for information on the query statements.

Batch Simulator Commands

This section contains the commands that are only available through the Batch Simulator. See Chapter 5, "System Administration" and the *VISION:Inform User Guide* for information on the use of the commands that are available through the online version, as well as the batch version of VISION:Inform.

COPY Command (Edit Mode)

The COPY command inserts a collection of statements (from another item in the foreground library) into the group of statements currently being edited. You can omit all parameters if the order shown is preserved.

Syntax:

COPY ITEM name TYPE { QUERY / PROFILE / STMTS } , INSERT line number LIST

Parameters:

ITEM name

Specifies the collection of statements to be copied (the collection has been stored previously with a SAVE command).

TYPE { QUERY / PROFILE / STMTS }

Specifies the type of collection being copied. If you omit the parameter, TYPE QUERY is assumed.

INSERT line number

Specifies where the statements are to be inserted. Statements following this point are renumbered if necessary. If you omit this entry, the statements are inserted after the last statement in the edit mode work area.

LIST

Specifies that the copied statements are to be listed after they are copied.

Examples:

```
COPY ITEM AUDIT INSERT 120
COPY FINANCE TYPE STMTS LIST
COPY YEARID
```

DELETE Command (Edit Mode)

The DELETE command in edit mode (EDIT:) removes one or more statements from the collection of statements being edited.

Syntax:

DELETE FROM number TO { **number** | **END** }

Parameters:

Note: *Entering a line number only, without any text, deletes the line with that number, if it exists.*

FROM number

The FROM parameter number indicates the first statement to be deleted. You can omit the FROM. The default is the first line.

TO { **number** | **END** }

The TO parameter number indicates the last statement to be deleted.
When you include the TO parameter, all statements from the FROM number up to and including the TO number or END are deleted.

- You can omit the TO.
- If you omit the FROM number and specify the TO number, all statements up to and including the TO number are deleted.
- END deletes every statement up to and including the last statement.

In the following examples, assume a set of statements, numbered 10, 20, 30, ..., 100:

Command	Result
DELETE FROM 10 TO 40	Deletes statements 10, 20, 30, and 40.
DELETE 40	Deletes statement 40.
DELETE 40 TO END	Deletes statements 40, 50, 60, 70, 80, 90, and 100.
DELETE 10 60	Deletes statements 10, 20, 30, 40, 50, and 60.
DELETE TO 60	Deletes all the statements from the first line up to and including statement 60.

40	Deletes statement 40. Entering a line number only, without any text, deletes the line with that number.

DELETE Command (System Mode)

In system mode (?:), the DELETE command removes an entire collection of statements from the foreground library.

Syntax:

Parameters:

ITEM name

Specifies the collection of statements to be deleted (the collection has been stored previously with a SAVE command). ITEM is a required parameter.

Specifies the type of collection being deleted. If you omit this parameter, TYPE QUERY is assumed.

Examples:

```
DELETE ITEM AUDIT
DELETE ITEM FIND TYPE STMTS
DELETE ITEM YEARID
```

EDIT Command (System Mode)

The EDIT command initiates the edit mode from the system mode.

Syntax:

Parameters:

ITEM name

Specifies the name of an existing query, profile, or collection of statements to be retrieved from the library for updating.

- The ITEM keyword is optional if the name follows the EDIT command.
- If you omit ITEM and name, a new item is created.

- QUERY indicates the statements will form a complete query and are to be validated for both syntax errors and internal consistency as entered.
- PROFILE indicates the statements will form a complete profile and are to be validated for both syntax errors and internal consistency as entered.
- STMTS indicates a partial query will be entered. The statements will be checked for syntax only.

If you omit the TYPE parameter, TYPE QUERY is assumed.

NOCHECK

Specifies that the statements are not to be validated.

Examples:

```
EDIT

EDIT ITEM LAB1 TYPE QUERY
```

END EDIT Command (Edit Mode)

The END EDIT command terminates the edit mode of operation, clears the edit mode work area, and places you in the system mode.

Syntax:

END EDIT

Insert/Replace Command (Edit Mode)

The insert/replace command inserts a statement in the statements being edited.

Syntax:

nn statement

Note: *If the statement number already exists, the statement is replaced.*

Parameters:

nn

Specify a number to place the statement in the correct sequential order.

statement

Specify the new statement to be inserted in the query, profile, or collection of statements.

LOGON Command

The LOGON command initiates a Batch Simulator session and identifies the user.

Syntax:

LOGON USERID userid PASSWORD pswd

Parameters:

USERID userid

Specifies the user ID. You can omit the USERID keyword if the order shown is preserved.

PASSWORD pswd

Specifies the password (if necessary). You can omit the PASSWORD keyword if the password is entered after the user ID.

Examples:

```
LOGON USERID DONNA PASSWORD GAIL
LOGON SYSTEM SECRET
```

RENUMBER Command (Edit Mode)

The RENUMBER command assigns new line numbers to the statements being edited.

Syntax:

RENUMBER FROM number BY number START number

Parameters:

RENUMBER

When you enter this command by itself, all statements in the query, profile, or collection of statements are renumbered using the defaults which follow.

FROM number

Specifies the line number at which renumbering begins. If the FROM keyword is omitted, renumbering begins at the first statement.

BY number

Specifies the value by which the line numbers are to be incremented. If you omit the BY keyword, BY 10 is assumed.

START number

Specifies the value to be assigned to the first number changed. If you omit the START keyword, the statement following the FROM value is used.

Examples:

```
RENUMBER BY 5
RENUMBER
RENUMBER FROM 80 BY 5 START 90
RENUMBER 10 5 5
```

SAVE Command (Edit Mode)

The SAVE command names and saves a query, profile, or collection of statements in the foreground library for future use.

Syntax:

When you enter the command SAVE alone, it replaces the item that was retrieved with the EDIT command. The item is saved under the same name.

Parameters:

ITEM name

Specifies the name of the query, profile, or collection of statements to be saved in the foreground library. You can omit the ITEM keyword, if you preserve the order shown. Start the item name with an alphabetic letter and make it 1 - 8 alphanumeric characters in length. Do not use the system delimiter.

Specifies the item as a query (QUERY), profile (PROFILE), or collection of statements (STMTS). If you omit this parameter, the type specified on the EDIT statement is assumed.

NOCHECK

Specifies that the statements are not to be validated.

The SAVE command operates in the following manner:

- If the item is not already in the foreground library, the item is stored with the specified name.
- If the item already exists in the foreground library, the item replaces what was stored in the library.

Examples:

```
SAVE

SAVE ITEM AUDIT TYPE QUERY NOCHECK

SAVE CHECKD TYPE STMTS

SAVE USER8 PROFILE
```

SUBMIT Command (Edit Mode and System Mode)

In edit mode (EDIT:), the SUBMIT command causes the query currently being edited to be submitted for execution (that is, run or initiate the query).

Syntax:

```
SUBMIT PRINT MAXITEMS value CLASS value MAXLINES value SOURCE,
       OUTFMT value OVERRIDE
```

In the system mode (?:), the SUBMIT command submits a previously saved, named query for processing.

Syntax:

```
SUBMIT   QUERY name USERID userid PRINT MAXITEMS value CLASS value,
         MAXLINES value SOURCE OUTFMT value OVERRIDE
```

Parameters:

QUERY name

In system mode (?:), you specify the QUERY keyword, followed by the query name that you want to submit. The QUERY keyword is optional, if the query name immediately follows SUBMIT. The specified query name must exist in the foreground library.

When you are in edit mode (EDIT:), QUERY name is not required, because you are submitting the query you are editing.

USERID userid

In system mode, specifies the user ID of the owner of the stored query. This parameter is not required unless the query being submitted belongs to a user ID different from the user ID submitting the query.

PRINT

Optional parameter to route the output to the system printer. If you omit this parameter, the output is stored in the communication file.

MAXITEMS value

An optional parameter that indicates the number of data records to be processed by this query. Maximum is 8 digits.

CLASS value

A number 1 through 15 specifying the class of the query. Queries are batched and executed according to class and the schedule established by the system administrator.

- The Batch Simulator routes classes 1 - 14 to the communication file where they are batched for processing by a Background Processor.
- The Batch Simulator immediately processes class 15 and routes the output directly to the system printer. The query is not stored in the communication file.

If you do not specify CLASS, VISION:Inform uses the default class from your user profile or the PARMBLK.

MAXLINES value

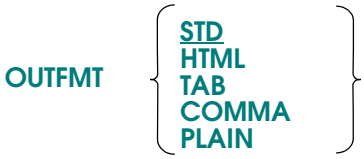
An optional parameter that indicates the number of lines of output the query is to print. Use this to limit output when testing a query. Maximum is 4 digits.

SOURCE

An optional parameter that prints the query source statements preceding the first report page.

OUTFMT value

An optional parameter that indicates the desired output format of the report for the submitted query. The possible values for OUTFMT are::



The acceptable values for OUTFMT are:

- **STD** indicates that the output is to be idn standard report format, which is the default.
- **HTML** indicates that the output is to be in HTML (**H**yper**T**ext **M**arkup **L**anguage).
- **TAB** indicates that the output is to be in tab-delimited format.
- **COMMA** indicates that the output is to be in comma-delimited format (also known as **C**omma **S**eparated **V**ariable, or CSV).
- **PLAIN** indicates that the output is to be in plain text format.

The OUTFMT keyword applies to all reports in a submitted query that do not already have an output format specification on the FORMAT statement in the respective REPORT group.

OVERRIDE

An optional parameter that applies to the output format specified on the OUTFMT keyword of the SUBMIT command. When you specify OVERRIDE on the SUBMIT command in conjunction with OUTFMT, the specified OUTFMT will override any report output format that may have been specified in the query source on the FORMAT statement.

Edit Mode Examples:	**System Mode Examples:**
SUBMIT	SUBMIT ACCOUNTS CLASS 10
SUBMIT MAXLINES 100	SUBMIT QUERY EMPLOYEE MAXITEMS 50

SYSTEM Command (Edit Mode)

The SYSTEM command switches to the system mode (?:) from the edit mode (EDIT:).

The edit mode work area is not affected. (See END EDIT).

Syntax:

SYSTEM

Parameter:

None.

Sample Input Statements

This section contains examples of Batch Simulator input statements. Explanations follow each example.

```
1-->  LOGON SYSTEM

2-->  EDIT TYPE QUERY

3-->  QUERY DATABASE CUSTOMER

3-->  REPORT CUSTNO CUSTNAME

3-->  TITLE 'LIST CUSTOMER FIELDS'

3-->  END REPORT

3-->  END QUERY

4-->

5-->  SAVE ITEM QUERY1 TYPE QUERY

6-->  SUBMIT

7-->  QUIT
```

Figure 6-2 Sample Batch Simulator Input statements

The following explains the numbered statements in Figure 6-2.

Line no.	Explanation
1	Log on to the Batch Simulator. You are automatically placed in system mode.
2	Switch to edit mode to create a query.
3	The query statements.
4	Switch back to system mode.
5	Save the query in the foreground library under the name QUERY1.
6	Submit the query for execution.
7	End the Batch Simulator session.

The following figure, Figure 6-3, is a sample of the output you would receive if you entered the LIST command from edit mode after creating the query in Figure 6-2.

```
LIST

10 QUERY DATABASE CUSTOMER

20 REPORT CUSTNO CUSTNAME

30 TITLE 'LIST CUSTOMER FIELDS'

40 END REPORT

50 END QUERY
```

Figure 6-3 LIST Command Output

Based on the output from the LIST command, you can use the insert/replace, DELETE, LIST, and RENUMBER commands. Figure 6-4 shows an example of how you would use some of these commands based on the output shown in Figure 6-3.

```
1--> LOGON SYSTEM

2--> EDIT QUERY1 TYPE QUERY

3--> 20 REPORT CUSTNO CUSTNAME ORDERNO INAMT

4--> 25 FORMAT DATEPOS NO TITLEPOS BOTTOM

5--> LIST

6--> RENUMBER

7--> SAVE QUERY2 TYPE QUERY

8--> QUIT
```

Figure 6-4 Sample Batch Simulator Statements

An explanation of the numbered statements in Figure 6-4 follows:

Line no.	Explanation
1	Log on to the Batch Simulator.
2	Edit the existing query named QUERY1.
3	Replace line 20 with this new statement.
4	Add line 25 which is a FORMAT statement.

Line no.	Explanation
5	List the contents of the query being edited. The contents are: 10 QUERY DATABASE CUSTOMER 20 REPORT CUSTNO CUSTNAME ORDERNO INAMT 25 FORMAT DATEPOS NO TITLEPOS BOTTOM 30 TITLE 'LIST CUSTOMER FIELDS' 40 END REPORT 50 END QUERY
6	Renumber the query. The contents are now: 10 QUERY DATABASE CUSTOMER 20 REPORT CUSTNO CUSTNAME ORDERNO INAMT 30 FORMAT DATEPOS NO TITLEPOS BOTTOM 40 TITLE 'LIST CUSTOMER FIELDS' 50 END REPORT 60 END QUERY
7	Save the query under a new name QUERY2. QUERY1 is left unchanged.
8	End the Batch Simulator session.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y