

# **FOCUS for S/390**

Simultaneous Usage Reference Manual  
CMS Version

---

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>1-1</b>
	What Is Simultaneous Usage? .....	1-2
	How SU Processes Transactions .....	1-3
<b>2</b>	<b>Operating the Sink Machine.....</b>	<b>2-1</b>
	Starting a Sink Machine Using VMCF.....	2-2
	Using the SU Profile .....	2-3
	Increasing the Maximum Number of Users .....	2-4
	Stopping a Sink Machine.....	2-4
	Logging Activity on the HLIPRINT File .....	2-7
	HLIPRINT File Contents .....	2-8
	HLIPRINT File Contents (Extended Format) .....	2-10
	Creating and Using an HLIPRINT/HLIERROR File.....	2-12
	Writing Reports With the Extended HLIPRINT Trace .....	2-14
	The Mult3-Threaded HLI/SU Reporting Facility .....	2-16
<b>3</b>	<b>SU and the FOCUS Language.....</b>	<b>3-1</b>
	Using Centrally Controlled Databases.....	3-2
	Messages From the Sink Machine.....	3-3
	Error Messages .....	3-3
	Listing Users on the Sink Machine: The ? SU Query .....	3-4
	MODIFY and MAINTAIN Facility Considerations .....	3-6
	Indicating Conflicts: The FOCURRENT Variable.....	3-7
	Testing for Rejected Transactions .....	3-8
	Validating CRTFORM Turnaround Fields.....	3-9
	Protecting Sink Transactions .....	3-12
	The Checkpoint Facility .....	3-12
	Managing MODIFY and MAINTAIN Transactions: COMMIT and ROLLBACK .....	3-13
	Performance Considerations .....	3-18
	Comparing COMMIT and ROLLBACK to the Checkpoint Facility .....	3-18
	Using FOCURRENT .....	3-18
	Seeing Other Users' Changes While Using COMMIT and ROLLBACK .....	3-19
	Using Commands in the SU Profile for COMMIT and ROLLBACK .....	3-19
	Controlling COMMIT Processing on the Sink Machine .....	3-19
	Controlling Currency Checking on the Sink Machine .....	3-20

<b>4</b>	<b>SU and the Host Language Interface (HLI)</b> .....	<b>4-1</b>
	Creating an HLI Module .....	4-2
	The File Communication Block (FCB) .....	4-2
	FCB Area Layout .....	4-3
	Using Both Local and Centrally Controlled Databases .....	4-4
	Error Codes .....	4-4
	Closing Centrally Controlled Databases.....	4-5
	Writing Transactions From the Buffer .....	4-5
	Stopping a Sink Machine Using HLI Control Commands .....	4-5
	Communications Protocol .....	4-6
<b>5</b>	<b>Inter User Communication Vehicle (Release 7.0.8 and Up)</b> .....	<b>5-1</b>
	Starting a Sink Machine Using IUCV .....	5-2
	VM Command Line Support .....	5-3
	Server Commands .....	5-4
	Server Query Commands .....	5-5
	Server Parm Commands .....	5-6
	Configuration and Data Files.....	5-7
	CP SMSG Support.....	5-8
	NUCXLOAded in Storage .....	5-8
	CP TRACE MC.....	5-9
	Segment TRACE.....	5-9
	Automatic IPL .....	5-10
	FULL and SNAP Dumps .....	5-10
	CP MONITOR .....	5-11
	User Exits .....	5-11
	Security Considerations.....	5-12
	Special Passwords .....	5-12
	Issuing Commands Without a Password .....	5-12
	FOCSUACC DATA.....	5-13
	EXEC Services .....	5-14
	SUBCOM for EXECs .....	5-15
	File Checkpoint .....	5-15
	LOADLIB GLOBAL .....	5-15
	Unsolicited Interrupts .....	5-16
	FILEDEFs .....	5-16
	Sample Directory.....	5-17

<b>A</b>	<b>Improving Performance: Storing Central Databases On Disk.....</b>	<b>A-1</b>
<b>B</b>	<b>FOCUS Error Messages.....</b>	<b>B-1</b>
<b>C</b>	<b>HLI Status Codes Returned in FCB Word 24 .....</b>	<b>C-1</b>
<b>Index</b>	<b>.....</b>	<b>I-1</b>

Cactus, EDA, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, Information Builders, the Information Builders logo, SmartMode, SNAPpack, TableTalk, and Web390 are registered trademarks and Parlay, SiteAnalyzer, SmartMart, and WebFOCUS are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

NOMAD is a registered trademark of Aonix.

UniVerse is a registered trademark of Ardent Software, Inc.

IRMA is a trademark of Attachmate Corporation.

Baan is a registered trademark of Baan Company N.V.

SUPRA and TOTAL are registered trademarks of Cincom Systems, Inc.

Impromptu is a registered trademark of Cognos.

Alpha, DEC, DECnet, NonStop, and VAX are registered trademarks and Tru64, OpenVMS, and VMS are trademarks of Compaq Computer Corporation.

CA-ACF2, CA-Datcom, CA-IDMS, CA-Top Secret, and Ingres are registered trademarks of Computer Associates International, Inc.

MODEL 204 and M204 are registered trademarks of Computer Corporation of America.

Paradox is a registered trademark of Corel Corporation.

StorHouse is a registered trademark of FileTek, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Informix is a registered trademark of Informix Software, Inc.

Intel is a registered trademark of Intel Corporation.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, IBM, MQSeries, MVS, OS/2, OS/400, RACF, RS/6000, S/390, VM/ESA, and VTAM are registered trademarks and DB2/2, HiperSpace, IMS, MVS/ESA, QMF, SQL/DS, VM/XA and WebSphere are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbit is a registered trademark of Iona Technologies Inc.

Approach and DataLens are registered trademarks of Lotus Development Corporation.

ObjectView is a trademark of Matesys Corporation.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

NetWare and Novell are registered trademarks of Novell, Inc.

CORBA is a trademark of Object Management Group, Inc.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

PeopleSoft is a registered trademark of PeopleSoft, Inc.

INFOAccess is a trademark of Pioneer Systems, Inc.

Progress is a registered trademark of Progress Software Corporation.

Red Brick Warehouse is a trademark of Red Brick Systems.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

Silverstream is a trademark of Silverstream Software.

ADABAS is a registered trademark of Software A.G.

CONNECT:Direct is a trademark of Sterling Commerce.

Java, JavaScript, NetDynamics, Solaris, and SunOS are trademarks of Sun Microsystems, Inc.

PowerBuilder and Sybase are registered trademarks and SQL Server is a trademark of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2000, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

# Preface

The Simultaneous Usage facility (SU) enables several users to read and maintain a FOCUS database at the same time by using the FOCUS language or by using other programming languages through the Host Language Interface (HLI).

This document describes how to operate the SU facility in CMS, both from FOCUS and from HLI. It is assumed that you are familiar with CMS and FOCUS (especially the MODIFY and MAINTAIN commands) or HLI.

## How This Document is Organized

- Chapter 1, *Introduction*, introduces the Simultaneous Usage facility and explains how it controls databases.
- Chapter 2, *Operating the Sink Machine*, describes the sink machine (starting and stopping, messages, and the HLIPRINT log file).
- Chapter 3, *SU and the FOCUS Language*, discusses working in SU with the FOCUS language.
- Chapter 4, *SU and the Host Language Interface (HLI)*, discusses working in SU with HLI. For more information on HLI, see the *Host Language Interface Users Manual*.
- Chapter 5, *Inter User Communication Vehicle (Release 7.0.8 and Up)*, discusses the communication layer for the new HLI IUCV server.

**Note:** The Inter User Communication Vehicle (IUCV) is only available with FOCUS Release 7.0.8 and higher.

The appendices include:

- Technical notes.
- FOCUS error messages and explanations.
- HLI return codes and explanations.

## Documentation Conventions

The following conventions are used throughout this manual:

<code>THIS TYPEFACE</code>	Denotes a term that you must enter exactly as shown.
<code>this typeface</code>	Denotes a term for which you must substitute an appropriate value.
<code>underscore</code>	Indicates a default option.
<code>{ }</code>	Indicates a required group of parameters; you must select one.
<code>[ ]</code>	Indicates an optional group of parameters; none is required.
<code>...</code>	Indicates that you can enter a term or parameter multiple times.
<code>.</code>	Indicates that there may be intervening commands.
<code>.</code>	
<code>.</code>	

## User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. Thank you, in advance, for your comments.

## Customer Support

Questions about FOCUS? Call Information Builders Customer Support Service (CSS) at (212) 736-6130 or visit our World Wide Web site (<http://www.ibi.com>). Customer service representatives are available between 8:00 a.m. and 8:00 p.m. EST to address all your FOCUS questions.

You can also upload your questions to the FOCWIZARD or FOCSERVICES forums on CompuServe. We do not accept questions via fax.

Information Builders consultants can also give you general guidance on FOCUS capabilities and documentation.

## **Information Builders Consulting and Training**

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products. For more information, course descriptions, locations, and dates, or to register for classes, visit the Information Builders Education Emporium at our World Wide Web site (<http://www.ibi.com/training>) or call 1-800-969-INFO to speak to an Education Representative.

---

## CHAPTER 1

# Introduction

### Topics:

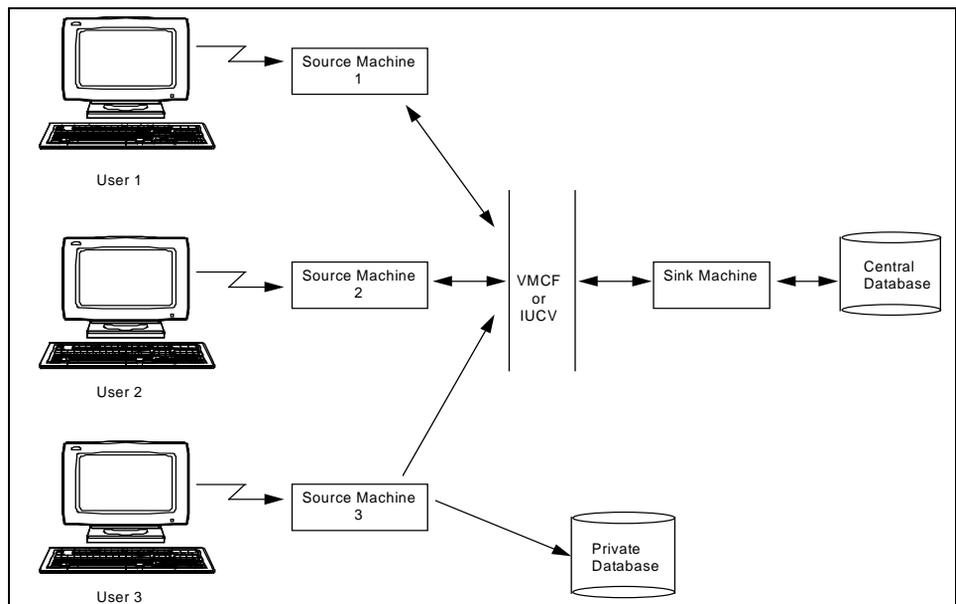
- What Is Simultaneous Usage?
- How SU Processes Transactions

This chapter introduces the concept of Simultaneous Usage (SU). The first section explains what Simultaneous Usage is; the second section explains how it works.

# What Is Simultaneous Usage?

Many users can read and change the same FOCUS database at the same time, through both FOCUS and Host Language Interface (HLI) commands, by using a facility called Simultaneous Usage. In SU, the database is called a “centrally controlled database.” It exists on a separate user ID called a “sink machine.” The users’ CMS IDs that are running FOCUS or HLI programs are called “source machines.” The users (via their source machines) send requests and transactions to the sink machine. The sink machine processes the transactions and transmits the retrieved data or messages back to the source machine.

Figure 1-1 depicts the SU environment. Three users (source machines) are executing FOCUS requests. The source machines communicate with the sink machine through the Virtual Machine Communication Facility (VMCF) or the Inter User Communication Vehicle (IUCV). When the sink machine receives a request from a source machine, it changes or retrieves data from the centrally controlled database. The sink machine then transmits the results back to the source machine. Notice that User 3 is also working on a locally controlled database, which is not under SU control.



**Figure 1-1. Components of the Simultaneous Usage Environment**

SU is not necessary if all users are just reading a database (for example, if they enter TABLE requests). SU is needed, however, if one user wants to modify the database while other users are reading it. In that case, all users, even those just reading the database, must use the database in the SU environment.

With the Multi-Threaded SU Reporting Facility, users can read a centrally controlled database without using a sink machine. Chapter 3, *SU and the FOCUS Language*, explains how to use the Multi-Threaded SU Reporting Facility.

**Note:**

- Up to 256 source machines (users) can modify a single database in SU at once.
- IUCV has a limit of 2048 users or source machines.
- One source machine can communicate simultaneously with up to eight sink machines.
- Releases beginning with 657/9302 and up may communicate with any higher-release sink machine.

## How SU Processes Transactions

When you submit a transaction to modify a segment instance (with a MODIFY or MAINTAIN request or an HLI program), SU follows a procedure called “change verify protocol.” If two or more users try to modify the same segment instance at the same time, the change verify protocol determines which transaction is accepted. The procedure is as follows:

1. You first identify the instance to be changed. In FOCUS, you do this with MATCH or NEXT statements within a MODIFY or MAINTAIN request. In HLI, you do this with NEX or locate commands within your program.
2. Your source machine forwards the identified values to the sink machine, which uses the values to retrieve the correct instance.
3. The sink machine retrieves the original data file instance, holds one copy, and sends another to the source (userid) that requested the data.
4. Your MODIFY/MAINTAIN request or HLI program indicates what changes are to be made to the instance. Your source machine updates its copy of the instance with the new field values or marks the copy for deletion.
5. Your source machine sends the updated copy back to the sink machine.
6. The sink machine compares the copy of the instance that it saved with the instance stored in the data file to check whether the data file instance has since been updated by another user. At this point, two courses of action are possible:
  - If the copy and the current instance in the data file are the same, FOCUS changes the instance using the copy from the source machine.
  - If the original and the current instance in the data file are different, SU signals a conflict and rejects the source machine copy.

Notice that a source machine may work on separate, locally controlled databases.

---

## Sample FOCUS MODIFY Request

The following example of a FOCUS MODIFY request illustrates how change verify protocol works. The same concepts apply to HLI programs as well.

```
MODIFY FILE EMPLOYEE
PROMPT EMP ID PAY DATE
MATCH EMP ID
    ON NOMATCH REJECT
    ON MATCH CONTINUE
MATCH PAY DATE
    ON NOMATCH REJECT
    ON MATCH PROMPT GROSS
    ON MATCH UPDATE GROSS
DATA
```

When you execute this request, FOCUS prompts for an employee ID and pay date. You enter:

```
    EMP_ID    =
071382660
    PAY_DATE  =
820831
    GROSS    =
1050.35
```

The source machine forwards the employee ID value to the sink machine. The sink machine retrieves the instance with this ID and saves it. The sink machine also sends a copy to the source machine. After the source machine receives its copy, the source machine forwards the pay date to the sink machine, which retrieves this segment instance where PAY\_DATE is 820831 and GROSS is 916.67.

The sink machine saves these values for itself. It also sends another copy to your source machine. The source machine then prompts you for a new GROSS value, to which you respond:

```
GROSS=
1050.35
```

The source machine then updates its copy of the instance with the new GROSS value, \$1050.35.

Then the updated copy is sent back to the sink machine. At this point, the sink machine needs to verify that another user, in the meantime, has not modified the database. When several users try to change an instance in a centrally controlled database at the same time, FOCUS accepts the first transaction submitted and rejects the others. The other users are not aware of the changes the first user made, and might not want to change the instance if they were. They can repeat their transactions later.

Assume, for this example, that another user has changed the database using the same MODIFY request. The sink machine compares the original value of 916.67 to the value currently in the database. Since another user has already changed the database value, these values do not match, your transaction is rejected, and an error message displays on your terminal. After receiving the error message, you can resubmit your transaction. Enter the same key values again for a fresh copy of the instance.

## Sample FOCUS MAINTAIN Request

When using MAINTAIN to write to databases controlled by a sink machine, your request can access only databases that reside on that sink. Consider the following example:

```

MAINTAIN FILE EMPLOYEE
1. FOR ALL NEXT EMP_ID PAY_DATE GROSS INTO EMPSTK
   COMPUTE TROWS/I4 = EMPSTK.FOCCOUNT;
2. WINFORM SHOW EMPGRID
3. CASE UPD
   FOR TROWS UPDATE SALINFO.GROSS FROM EMPSTK
   ENDCASE
.
.
.

```

1. The FOR ALL NEXT command retrieves all records from the database and places each instance of the path beginning with EMPINFO, FUNDTRAN, PAYINFO, ADDRESS, SALINFO segments into the EMPSTK stack.
2. The WINFORM command displays the EMPGRID Winform. It shows the EMP\_ID, PAY\_DATE, and GROSS fields in a grid.
3. This case will update all the GROSS records that were placed in the EMPSTK stack.

**Note:** FOCUS provides two MODIFY and MAINTAIN subcommands, COMMIT and ROLLBACK, which enable you to control when changes are made to the sink machine database. COMMIT and ROLLBACK are discussed in Chapter 3, *SU and the FOCUS Language*.

---

## CHAPTER 2

# Operating the Sink Machine

### Topics:

- Starting a Sink Machine Using VMCF
- Stopping a Sink Machine
- Logging Activity on the HLIPRINT File
- The Multi-Threaded HLI/SU Reporting Facility

This chapter describes how to operate the sink machine. The chapter is addressed to system and database administrators and others who want to operate a sink machine. It discusses the following topics:

- How to start and stop the sink machine (the user ID which controls the centrally controlled databases).
- How to log its activity in a sequential file.
- How to write summary reports on sink machine activity.
- How to use the Multi-Threaded HLI/SU Reporting Facility.

Once the sink machine is started, it needs no more attention until the user wants to stop it. After it is started, users may read and modify the databases under its control.

---

## Starting a Sink Machine Using VMCF

To begin SU, you or your system administrator must log onto the sink machine user ID, load the program HLIMAIN that runs SU, and disconnect the terminal from the sink machine. To do this, logon and enter the command

```
FOCUS SU [password] [ECHO|STAT]
```

**Note:** To do this with IUCV, refer to Chapter 5, *Inter User Communication Vehicle (7.0.8 and Up)*.

where:

*password*

Is an optional string of up to eight characters. The password is important, only to HLI users, for remotely stopping the sink machine (see Chapter 4, *SU and the Host Language Interface (HLI)*). However, you must enter a password if you want to use the ECHO or STAT parameter; otherwise, FOCUS interprets the ECHO or STAT parameter as the password.

ECHO

STAT

Are optional parameters that cause FOCUS to record sink machine activity in the file which you FILEDEF to ddname HLIPRINT. *HLIPRINT File Contents (Extended Format)* on page 2-10 describes how to set up the HLIPRINT file.

FOCUS SU

Is a command that executes a CMS EXEC called FOCUSSU. This EXEC contains the following commands:

<code>FILEDEF SYSIN</code>	Needed for internal purposes.
<code>TERM</code>	
<code>CP SET RUN ON</code>	Allows the HLIMAIN program to run after the terminal has been disconnected from the sink machine.
<code>CP DISCONN</code>	Disconnects the terminal from the sink machine which will then execute the HLIMAIN program.
<code>HLIMAIN arg1 arg2</code>	Loads HLIMAIN. The arg1 variable is the password; the arg2 variable is either the ECHO or the STAT parameter.

You can write your own EXEC to start the sink machine, incorporating the commands above but also including other commands that assist in using SU. Information Builders ships a sample REXX EXEC that can be used to start your VMCF or IUCV SU machine. Please refer to the FOCUSSU EXEC on your production disk.

## Using the SU Profile

The Simultaneous Usage Profile (SU Profile) enables you to set several parameters for the FOCUS Simultaneous Usage Machine (sink machine) in a profile. These parameters can also be used to control a local HLI program.

The profile is named PROFILE HLI, with LRECL 80 and RECFM F. It should reside on a minidisk that the sink machine accesses.

You can include the following commands in the sink machine profile:

```
SET BINS = nn
SET CACHE = nn
SET PATHCHECK = ON/OFF
```

where:

### SET BINS

Controls the number of I/O buffers for the sink machine. A maximum of 64 bins (which are shared by all users) is allowed. Set BINS to the maximum for optimal performance. If BINS are not set in the profile, the sink calculates how many BINS to allocate based on the available storage.

Note that source users can use the SET BINS command for their own processing. If the SET BINS command is included in the sink machine profile, it only affects Simultaneous Usage processing.

### SET CACHE

Cache memory buffers FOCUS database pages between disk and BINS and reduces I/O to disk. Using the SET CACHE command in the profile keeps the entire database in memory and, therefore, improves performance. The default is no cache memory. See Chapter 21 of the *FOCUS Users Manual* for more information on using cache memory with FOCUS Files.

Note, as with the SET BINS command, source users can use the SET CACHE command for their own processing. If the SET CACHE command is included in the sink machine profile, it only affects Simultaneous Usage processing.

### SET PATHCHECK

Enables you to control whether the sink machine determines the value of FOCURRENT by checking an updated segment and its parent segments in the path (SET PATHCHECK = ON, default), or by checking only the updated segment (SET PATHCHECK = OFF). The SET PATHCHECK command can be used only when COMMIT=ON for the sink machine. SET PATHCHECK and the FOCURRENT variable are discussed in Chapter 3, *SU and the FOCUS Language*.

The commands read from the profile and any errors are written to ddname HLIERROR. If HLIERROR is not allocated, the commands and errors will be written to HLIPRINT. If neither HLIERROR or HLIPRINT is allocated, errors are written to the terminal.

---

The following error message applies specifically to the SU profile:

```
(FOC725) ERROR IN HLI PROFILE
```

Only certain SET commands may be issued in the HLI profile. Either a non-SET line was found, or an invalid SET command was issued. Check the file (PROFILE HLI on CMS or member HLIPROF of the FOCEXEC dataset on MVS). The erroneous line was ignored.

## Increasing the Maximum Number of Users

A sink machine has a table that contains the count of the number of active FOCUS users accessing it. The default maximum is 1024 users. To decrease the 1024-user limit, edit the file SESSCM ASSEMBLE on the FOCUS maintenance disk and change the line that reads as follows:

```
NUMSES      EQU      1024
```

Replace the number 1024 with your new number of users; for example, 256.

Do not change any other lines in this program. After you make the change to NUMSES, file SESSCM ASSEMBLE back onto the FOCUS maintenance disk.

Assemble SESSCM with the following syntax:

```
ASSEMBLE SESSCM
```

**Note:** When the number of source machines reaches the maximum allowed, the source machine will issue the following error message:

```
(FOC542) SU. COMMUNICATION NOT AVAILABLE TO CENTRAL DATABASE MACHINE
```

The HLIPRINT file will contain the following message:

```
(FOC528) SU. TOO MANY FILES OPEN ON THE CENTRAL DATABASE MACHINE
```

## Stopping a Sink Machine

When preparing to stop a sink machine, first logon to it. When the RECONNECT message appears, press the ENTER key. The sink machine stops processing and the following message appears:

```
HLIMAIN INTERRUPTED ... ENTER HX,STOP,CALLS ? :
```

You then enter one of three replies:

- |                    |  |
|--------------------|--|
| <code>?</code>     | Displays a list of sink machine users.   |
| <code>STOP</code>  | Shuts down the sink machine normally.    |
| <code>HX</code>    | Shuts down the sink machine immediately. |
| <code>CALLS</code> | Displays the count of HLI calls.         |

Each of these replies is explained below.

**Note:** If you are using HLI, you can also shut down the sink machine from a source machine. See Chapter 4, *SU and the Host Language Interface (HLI)*.

## The ? Reply

The ? reply displays the source machine user IDs and the file IDs of the sink machine databases they are accessing. For example:

```

USERID  FILEID          QUEUE
JBP     CAR        FOCUS A    Q
NNA     CAR        FOCUS A

```

This list shows that user IDs JBP and NNA are now accessing the file CAR FOCUS A. The Q in the QUEUE column indicates that JBP has requests pending. (You can also obtain this list from a source machine if you are using the FOCUS language. See Chapter 3, *SU and the FOCUS Language*.)

Before you shut down the sink machine, display this list to determine if users are still accessing the sink machine. After this list appears, the sink machine resumes processing, allowing users who were interrupted to continue from the point they left off. You can then either shut down the sink machine or disconnect the terminal and allow SU mode to continue. Both alternatives are explained on the next page.

## The STOP Reply

The STOP reply allows the current queue of pending source machine requests to be processed and refuses any new ones. It then closes down normally and returns control to CMS.

**Note:** If you previously entered the ? reply described above, you must press ENTER to redisplay the HLIMAIN INTERRUPTED message, then enter STOP.

---

## The HX Reply

The HX reply immediately terminates the sink machine without processing any of the source machine requests currently in queue or writing the records in the buffers to the database. Since this reply abandons the sink machine, enter HX only if you are unable to shut down the sink machine with the STOP reply.

**Note:** If you previously entered the ? reply described above, you must press ENTER to redisplay the HLIMAIN INTERRUPTED message, then enter HX.

## Allowing SU Mode to Continue

If you logon to the sink machine to shut it down and then decide not to, you can disconnect the terminal from the sink machine and leave the sink machine running. The procedure for doing this depends on when you decide to disconnect:

- If you logged on but did not yet press ENTER, causing the HLIMAIN INTERRUPTED message to appear, simply enter the command:  

```
#CP DISCONN
```
- If you logged onto the sink machine and pressed ENTER, causing the HLIMAIN INTERRUPTED message to appear, first enter the reply ? to display the list of current users, then enter the #CP DISCONN command.

If you enter the #CP DISCONN command immediately after the HLIMAIN message appears, the source machines may hang in wait state when they try to use the sink machine. To remedy this:

1. Log back onto the sink machine.
2. When the RECONNECT message appears, enter the ? reply.
3. After the list of current users is displayed, enter the #CP DISCONN command. The source machines hanging in wait state will then resume processing.

When using a line terminal, do not enter the #CP DISCONN command after the HLIMAIN INTERRUPTED message appears. Instead, shut down the sink machine by entering STOP and restart it using the FOCUS SU command. If you entered the #CP DISCONN command by mistake, log back on and shut down the sink machine.

## Logging Activity on the HLIPRINT File

FOCUS can record all user actions on the sink machine in a sequential file called HLIPRINT. Each HLI command is a single action. Each MODIFY transaction, MAINTAIN transaction, or TABLE request may execute a series of actions such as opening and closing databases, reading databases, matching keys, and changing values. When you use the ECHO option, ddname HLIPRINT records:

- Each user action.
- The database on which the action took place.
- The database segment read or modified by the action.
- The user ID that issued the action.

In addition, FOCUS offers an extended form of the HLIPRINT file when you use the STAT option. In addition to the information displayed by the ECHO option, the extended form lists the following:

- Date and time of the action.
- The CPU time it took to execute the action.
- The number of I/O operations required to execute the action.
- The name of the FOCUS procedure (FOCEXEC) executing the action.
- For MODIFY and MAINTAIN requests with Case Logic, the name of the case executing the action.

The HLIPRINT file has three uses:

- It provides a guide for restarting procedures that end prematurely. The restart must be tailored to the application, because the HLIPRINT file does not log transaction data.
- It provides a means for performing usage accounting based on the number of actions.
- It provides a means for monitoring database use, particularly during testing of new procedures.

**Note:** If you display the HLIPRINT file on a system editor, the first column is reserved for print control characters.

## HLIPRINT File Contents

Figure 2-1 shows a sample HLIPRINT file obtained with the ECHO option. The columns are described on the next page.

CMD	FILENAME	STATUS	NEWSEG	TARGET	ANCHOR	NTEST	USERID	REF NUMB
OPN	FOCUSSU	FOCUS	F	0		DIW		00000001
RD	CAR	FOCUS	A	0		1 DIW		00000002
OPN	CAR	FOCUS	A	0		EFO		00000003
MINT	CAR	FOCUS	A	0		EFO		00000004
MATA	CAR	FOCUS	A	0	ORIGIN	EFO		00000005
MATB	CAR	FOCUS	A	0	COMP	EFO		00000006
MDEL	CAR	FOCUS	A	0	COMP	EFO		00000007
MATB	CAR	FOCUS	A	0	ORIGIN	NNR		00000008
MATB	CAR	FOCUS	A	1	COMP	NNR		00000009
MINC	CAR	FOCUS	A	0	COMP	NNR		00000010
MATB	CAR	FOCUS	A	0	ORIGIN	MHK		00000011
MATB	CAR	FOCUS	A	0	COMP	MHK		00000012
MDEL	CAR	FOCUS	A	0	COMP	MHK		00000013
MATB	CAR	FOCUS	A	0	ORIGIN	EFO		00000014
MATB	CAR	FOCUS	A	1	COMP	EFO		00000015
MINC	CAR	FOCUS	A	0	COMP	EFO		00000016
MATB	CAR	FOCUS	A	0	ORIGIN	NNR		00000017
MATA	CAR	FOCUS	A	1	COMP	NNR		00000018
MINC	CAR	FOCUS	A	0	COMP	NNR		00000019
CLO	CAR	FOCUS	A	0		NNR		00000020
CLO	FOCUSSU	FOCUS	*	0		NNR		00000021

**Figure 2-1. Contents of a Sample HLIPRINT File**

The columns in the chart are:

CMD

The type of action:

OPN	Open a file.
RD	Read a page from a FOCUS database.
MINT	MODIFY/SU initialization.
MATB MATA MATC MATD	Match a field value or values in the database (MODIFY MATCH subcommand).
MINC	Include a new segment instance (MODIFY INCLUDE subcommand).
MUPD	Update a segment instance (MODIFY UPDATE subcommand).
MDEL	Delete a segment instance and its descendants (MODIFY DELETE subcommand).

MNEX	Place the current position at the next segment instance in the segment chain (MODIFY NEXT subcommand).
MREP	Place the current position at the first segment instance in the segment chain (MODIFY REPOSITION subcommand).
MCMT	COMMIT subcommand.
MRBK	ROLLBACK subcommand.
NEW	Used by MAINTAIN.
SAV	Write transactions from buffer into the database. (How FOCUS writes transactions into the database is discussed in Chapter 3, <i>SU and the FOCUS Language</i> .)
CLO	Close a file.

Any other type of action in this column is an HLI command issued when using HLI with SU.

#### FILENAME

The CMS fileid (filename, filetype, and filemode) of the centrally controlled database on which the action took place. FOCUSSU FOCUS is a FOCUS work file. FOCUSSU FOCUS and FOCUSSU MASTER were provided at installation time.

#### STATUS

If the action executed normally, this value is set to 0. If a MATCH statement cannot locate a segment instance (ON NOMATCH condition), or if a NEXT statement reaches the end of a segment chain (ON NONEXT condition), this value is set to 1. Otherwise, the value is a FOCURRENT variable or an HLI status code. Chapter 3, *SU and the FOCUS Language*, lists FOCURRENT variable values; Appendix C, *HLI Status Codes Returned in FCB Word 24*, lists HLI status codes.

#### NEWSEG

For HLI only, the name of the first segment of new information.

#### TARGET

The segment in the FOCUS database read or modified by the action.

#### ANCHOR

For HLI only, the anchor segment.

#### NTEST

For HLI only, the number of test conditions qualifying a retrieved record.

For FOCUS users, the number of the database page that was read by a RD action (a page is a 4096-byte physical record of a FOCUS file).

#### USERID

The user ID that issued the action.

---

#### REF NUMB

The number of the action in the HLIPRINT file.

The REF NUMB number helps determine the last time a segment was changed. Use the FOCUS ? FILE command to display a chart showing when each segment in a FOCUS database was last changed. (This query is described in full in Chapter 21 of the *FOCUS Users Manual*.) The last column in the chart is labeled LAST TRANS NUMBER. This column lists a number for each segment in the file.

If the file was last modified under SU, the numbers in the LAST TRANS NUMB column will refer to the REF NUMB column in the HLIPRINT file, pointing to the last action performed on the segment (usually a CLO or SAV action).

## The HLI Maintain Command NEW

MAINTAIN SU has a new HLI command called NEW. This command provides a performance enhancement in the buffer size that is sent back from the SU server. It also receives arguments in the form of HLI commands. So when a database is opened with MAINTAIN for an update action, the NEW command will appear in HLIPRINT with an OPN command as an argument. This is shown in the following sample HLIPRINT file.

CMD	FILENAME	STATUS	NEWSEG	TARGET	ANCHOR	NTEST	USERID	REF NUMB	
OPN	FOCUSSU	FOCUS	A	0			PMSPAK	00000001	
RD	EMPLOYEE	FOCUS	A	0		1	PMSPAK	00000002	
RD	JOBFILE	FOCUS	A	0		1	PMSPAK	00000003	
RD	EDUCFILE	FOCUS	A	0		1	PMSPAK	00000004	
OPN	EMPLOYEE	FOCUS	A	0			PMSPAK	00000005	
NEW	EMPLOYEE	FOCUS	A	0	OPEN	00000048	00000312	PMSPAK	00000006

*Figure 2-2. Sample HLIPRINT File Containing the NEW Command*

## HLIPRINT File Contents (Extended Format)

Figure 2-3 shows a sample extended HLIPRINT file. The columns are the same as those in the simple HLIPRINT file described in the previous section, with the exception of the additions below.

**Note:** A Master File Description is supplied for the extended HLIPRINT file (called HLIPRINT MASTER) that allows you to write FOCUS reports from the file. Consult *Writing Reports With the Extended HLIPRINT Trace* on page 2-14 for examples.

Additional columns are:

#### DATE

Date the action was executed, in YYMMDD (year, month, day) format.

#### TIME

Time the action finished execution, in HHMMSS (hours, minutes, seconds) format.

**VTIME**

Amount of virtual CPU time required to execute the action. If the action took longer than 99.99 seconds to execute, VTIME displays a value of 99.99. Note that the value of VTIME is .0000 for the first action after the sink machine was started.

**TTIME**

The same as VTIME except it is the amount of total CPU time.

**IOS**

Number of database input/output (I/O) operations performed to complete the action. This number reflects only I/O operations on FOCUS databases and does not include I/O operations to other files such as Master File Descriptions. If the action required more than 9999 I/O operations, IOS displays a value of 9999.

**PROCNAME**

For FOCUS users, the name of the FOCUS procedure (FOCEXEC) executing the action. If the user is executing a FOCUS request live on the terminal, PROCNAME is blank.

For HLI users, the contents of FCB words 7 and 8. SU does not use words 7 and 8, so HLI users can define the contents of these words and have them displayed in the PROCNAME column.

**CASE NAME**

For MODIFY and MAINTAIN requests only.

If the request defines a temporary field called SUPRINTNAME (pronounced "SU print name"), CASE NAME is the value of this field. SUPRINTNAME must be an alphanumeric field no longer than 12 characters.

For example, if your MODIFY or MAINTAIN request contains the statement

```
COMPUTE SUPRINTNAME/A12 = 'MARK' ;
```

then the CASE NAME column will display the value MARK until the next COMPUTE statement places another value in the SUPRINTNAME variable.

If the request does not define SUPRINTNAME but does use Case Logic, CASE NAME is the name of the case that executed the action.

Otherwise, CASE NAME is blank.

This is a sample extended HLIPRINT file. A Master File Description is supplied for the extended HLIPRINT file that allows you to write FOCUS reports from the file.

1CMD	FILENAME	STATUS	NEWSEG	TARGET	ANCHOR	NTEST	USERID	REF NUMB	DATE	TIME	UTIME	TTIME	IOS	PROC NA
0														
RD	employee(focus)	0				1	HOFFMAN	00000001	950305	162254	.0000	.0000	1	
RD	jobfile(focus)	0				1	HOFFMAN	00000002	950305	162254	.0000	.0000	1	
RD	educfile(focus)	0				1	HOFFMAN	00000003	950305	162254	.0000	.0000	1	
OPM	employee(focus)	0				1	HOFFMAN	00000004	950305	162255	.0000	.0000	1	
MINT	employee(focus)	0					HOFFMAN	00000005	950305	162255	.0000	.0000	0	
MATC	employee(focus)	0		EMP INFD			HOFFMAN	00000006	950305	162319	.0000	.0000	0	
MATD	employee(focus)	0		SAL INFD			HOFFMAN	00000007	950305	162319	.0000	.0000	1	
MUPD	employee(focus)	0		SAL INFD			HOFFMAN	00000008	950305	162326	.0000	.0000	0	
CLD	employee(focus)	0					HOFFMAN	00000009	950305	162330	.0000	.0000	1	
RD	employee(focus)	0				1	HOFFMAN	00000010	950305	162347	.0000	.0000	1	
RD	jobfile(focus)	0				1	HOFFMAN	00000011	950305	162347	.0000	.0000	1	
RD	educfile(focus)	0				1	HOFFMAN	00000012	950305	162347	.0000	.0000	1	
RD	employee(focus)	0				1	HOFFMAN	00000013	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				2	HOFFMAN	00000014	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				3	HOFFMAN	00000015	950305	162348	.0000	.0000	1	
RD	jobfile(focus)	0				4	HOFFMAN	00000016	950305	162348	.0000	.0000	1	
RD	jobfile(focus)	0				1	HOFFMAN	00000017	950305	162348	.0000	.0000	1	
RD	jobfile(focus)	0				2	HOFFMAN	00000018	950305	162348	.0000	.0000	1	
RD	jobfile(focus)	0				3	HOFFMAN	00000019	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				4	HOFFMAN	00000020	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				5	HOFFMAN	00000021	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				6	HOFFMAN	00000022	950305	162348	.0000	.0000	1	
RD	educfile(focus)	0				3	HOFFMAN	00000023	950305	162348	.0000	.0000	1	
RD	educfile(focus)	0				2	HOFFMAN	00000024	950305	162348	.0000	.0000	1	
RD	educfile(focus)	0				1	HOFFMAN	00000025	950305	162348	.0000	.0000	1	
RD	employee(focus)	0				7	HOFFMAN	00000026	950305	162349	.0000	.0000	1	
RD	employee(focus)	0				7	HOFFMAN	00000027	950305	162350	.0000	.0000	1	
RD	employee(focus)	0				8	HOFFMAN	00000028	950305	162350	.0000	.0000	1	
RD	employee(focus)	0				5	HOFFMAN	00000029	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				3	HOFFMAN	00000030	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				2	HOFFMAN	00000031	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				1	HOFFMAN	00000032	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				2	HOFFMAN	00000033	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				3	HOFFMAN	00000034	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				2	HOFFMAN	00000035	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				1	HOFFMAN	00000036	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				2	HOFFMAN	00000037	950305	162350	.0000	.0000	1	
RD	educfile(focus)	0				3	HOFFMAN	00000038	950305	162350	.0000	.0000	1	

Figure 2-3. Sample Extended HLIPRINT File

## Creating and Using an HLIPRINT/HLIERROR File

To record activity in the HLIPRINT/HLIERROR file, follow these instructions when starting the sink machine:

1. If you are recording the HLIPRINT file on disk or tape, make sure that there is adequate storage space for the file. The HLIPRINT file can grow very quickly; a single MODIFY transaction, MAINTAIN transaction, or TABLE command can generate several actions and several lines of output. One day's use can produce many thousands of lines. If the space reserved for HLIPRINT becomes full, the sink machine cannot recover.
2. Define the HLIPRINT/HLIERROR file using the CMS FILEDEF command. If the file is new, give the file the following characteristics:
  - For simple HLIPRINT files (ECHO option): a fixed-record, blocked format, a record length of 88 characters. The blocksize must be a multiple of 88.
  - For extended HLIPRINT files (STAT option): a fixed-record, blocked format (RECFM FB), a record length of 133 characters. The blocksize must be a multiple of 133.

In general, the larger the blocksize, the fewer I/O operations FOCUS will perform. However, if the sink machine terminates abnormally, any records on the last unwritten block are lost.

For example, to record actions in a new file called SLOG DATA on the T disk of the sink machine, enter:

```
FILEDEF HLIPRINT DISK SLOG DATA T (RECFM FB LRECL 88 BLOCK 8800)
FILEDEF HLIERROR DISK SUERROR DATA T (RECFM FB LRECL 88 BLOCK 8800)
```

or

```
FILEDEF HLIPRINT DISK SLOG DATA T (RECFM FBA LRECL 133 BLOCK 13300)
FILEDEF HLIERROR DISK SUERROR DATA T (RECFM FBA LRECL 133 BLOCK 13300)
```

If the file exists, the new output will replace the present file contents. If you want to append the new output to the present file contents instead, add the DISP MOD parameter. For example, if you decide in a later session to append new output to the SLOG file contents, you would enter:

```
FILEDEF HLIPRINT DISK SLOG DATA T (DISP MOD)
```

If you start the sink machine with your own EXEC, you can include the FILEDEF command in the EXEC (see

---

*Starting a Sink Machine Using VMCF on page 2-2).*

The HLIPRINT file can be FILEDEFed to tape, to any disk to which the sink machine has write access, or to a virtual printer, reader, or punch.

3. Start the sink machine with the following command

```
FOCUS SU password {ECHO|STAT}
```

where:

`password`

Is a password up to eight characters long. You must enter a password, or FOCUS will interpret the following ECHO or STAT parameter as the password.

`ECHO`

Records actions in a simple HLIPRINT file.

`STAT`

Records actions in an extended HLIPRINT file.

If you start the sink machine with your own EXEC, add the password and the ECHO parameter to the line in the EXEC that invokes the HLIMAIN module. For example, if your EXEC is called SUSTART and your password is ABC, then SUSTART EXEC should contain the line:

```
HLIMAIN ABC ECHO
```

4. If you are using the extended HLIPRINT file and you want to print it, add the CC parameter to the CMS PRINT command. This enables the printer to interpret the control characters in column 1. For example:

```
PRINT SULOG DATA T (CC
```

## Writing Reports With the Extended HLIPRINT Trace

The extended HLIPRINT trace has a Master File Description provided with FOCUS called HLIPRINT MASTER. You can produce summary reports of your HLIPRINT trace with the FOCUSTABLE command. These summary reports may be used to:

- Determine peak sink machine usage by user, time of day, database, or application.
- Discover performance problems in procedures or in database design.
- Perform usage accounting.

Before issuing your report request, you must first FILEDEF the ddname HLIPRINT to the trace file. For example, if your HLIPRINT file was called SULOG DATA T, you would enter the following command from within FOCUS:

```
CMS FILEDEF HLIPRINT DISK SULOG DATA T (LRECL 133 RECFM FB BLOCK 13300
```

This sample report request determines the total amount of CPU and I/O operations used by the sink machine on behalf of each user:

```
TABLE FILE HLIPRINT
SUM VTIME AND TTIME AND IOS
BY USERID
END
```

Another report request could locate performance problems in an application:

```
TABLE FILE HLIPRINT
SUM VTIME AND TTIME AND IOS
BY PROC NAME BY CASE NAME
ON PROC NAME SUMMARIZE
END
```

The report displays as:

PROC NAME	CASE NAME	VTIME	TTIME	IOS
-----	-----	-----	-----	---
CARLOAD		.2177	.3204	6
*TOTAL CARLOAD		.2177	.3204	6
CARMOD	ADDCASE	.6730	1.1040	17
	DELCASE	.4426	.5919	10
	LOCATE	11.3587	19.8299	428
	SHOWCASE	.7341	1.2534	5
	TOP	.3906	.5745	6
	UPDATECASE	.6958	1.1790	21
*TOTAL CARLOAD		14.2948	24.5327	487
TOTAL		14.5125	24.8531	493

In this report, the case LOCATE in the procedure CARMOD consumes most of the CPU time and I/O resources. This particular part of the application might be redesigned to run more efficiently.

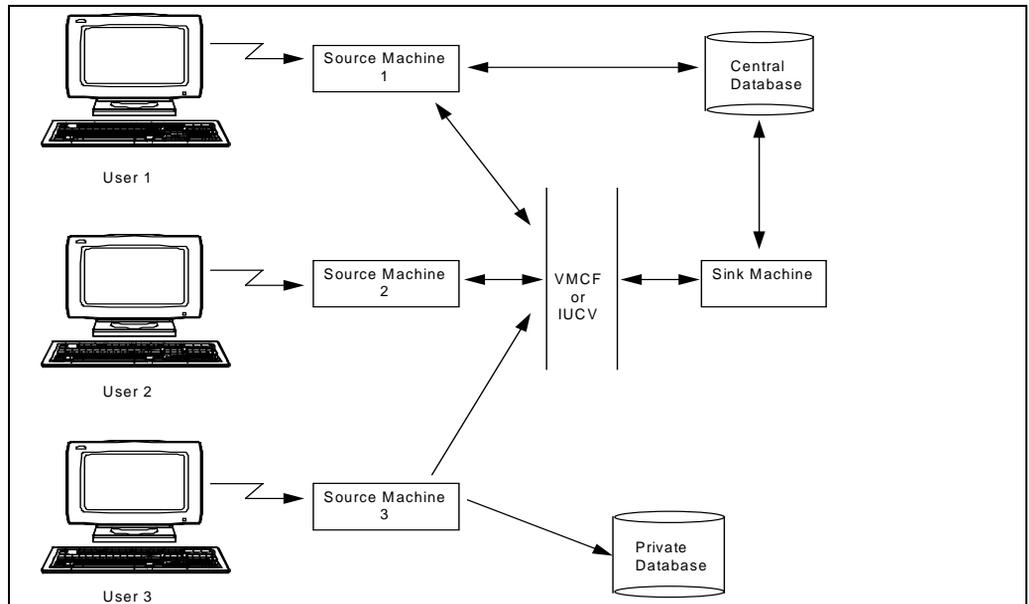
For more information about report requests, see Chapter 4 in your *FOCUS Users Manual*.

## The Multi-Threaded HLI/SU Reporting Facility

You can use the Multi-Threaded HLI/SU Reporting Facility to process report requests. With this facility, your source machine directly accesses the FOCUS database, bypassing the sink machine and the Virtual Machine Communication Facility (VMCF).

This Multi-Threaded Reporting Facility is available as of Release 5.5; and in FOCUS Release 6.5 and higher this feature extends to the HLI environment. Note that this feature is used in read-only mode and all modifications to the database are routed through the sink machine. In addition, a single multi-threaded SU program cannot open a file simultaneously in both read-only and read/write mode. Different FCBs must be written in order to report from and update the centrally controlled database.

In the following illustration, User 1 is using multi-threaded processing and has read-only access to the central database. Users 2 and 3 have read/write access to the central database via the sink machine.



**Figure 2-4. Components of the Multi-Threaded SU Reporting Facility**

With Multi-Threaded HLI/SU Reporting for CMS, the central database is accessed by each user as a local machine when performing operations and as a sink machine when performing modifications. In the HLI environment, the FCB is modified to denote the parallel configuration.

In the FOCUS environment, a USE command accomplishes this. Each database must be declared twice in the USE command for multi-threaded reporting. In the example below, although the CAR database and the SALES database are accessed on different sink machines, both are available for multi-threaded reporting.

```
USE
CAR FOCUS A ON sinkid1
CAR FOCUS B LOCAL
SALES FOCUS A ON sinkid2
SALES FOCUS C LOCAL
.
.
.
END
```

The keyword LOCAL indicates that the Multi-Threaded HLI/SU Reporting Facility will read the databases without going through their associated sink machines for TABLE requests. (MODIFY and MAINTAIN requests continue to go through sink machines as if this facility had not been used.) Note that you still must tell FOCUS what sink machine controls the database, even if you plan to read the database locally with the Multi-Threaded HLI/SU Reporting Facility.

To use the Multi-Threaded HLI/SU Reporting for CMS, you must perform the following steps:

1. Rename the SU FOCUS files with a filemode number of 6 (instead of the usual 1). For example, if a database resides on the sink machine as CARSU FOCUS A1, you must rename it as CAR FOCUS A6. Filemode number 6 causes CMS to rewrite FOCUS pages “in place,” overwriting previous pages, and is needed by the Multi-Threaded HLI/SU Reporting Facility because the source machine does not re-read the disk directory of the sink minidisk every time a new page is needed.

If the Absolute File integrity feature is desired, you must enter FOCUS, set SHADOW=ON, and rebuild the database. It is then renamed under the new filemode number.

In the example below, FOCUS shadow writing is specified, and the CAR file is rebuilt. The CAR filemode is renamed to filemode number 6.

```
SET SHADOW=ON
REBUILD
REBUILD
CAR FOCUS B1
NO
CMS RENAME CAR FOCUS B1 CAR FOCUS B6
```

---

**Note:** FOCUS shadow writing causes your file's space requirements to double. Be sure that adequate disk space and a backup copy of the file are available before starting the REBUILD process.

2. The source machine user must issue commands to link and access the central database as a local machine. For example, if the sink machine is running on user ID SINKMA, and the database is stored on disk 191, you could issue the following CMS commands to make the sink's 191 disk your local D disk:

```
CP LINK SINKMA 191 201 RR
ACCESS 201 D
```

3. Set up the FCB in the HLI program according to the following steps:
  - A. In FCB word 5 (FM), set byte 1 to the filemode of the sink machine and set byte 3 to the filemode of the local machine. Note that bytes 2 and 4 must remain as blanks.
  - B. Set FCB word 6 (SU) to SULO.
  - C. Set FCB words 9-10 (SINKID) to the user ID of the sink machine.

For example, assume the sink machine is running on user SINKMA, the central database disk is accessed locally as D and through the sink machine as A, and the following COBOL HLI FCB:

```
01 FCB
05 FCB-FN      PIC X(08)  VALUE SPACES.
05 FCB-FT      PIC X(08)  VALUE SPACES.
05 FCB-FM      PIC X(04)  VALUE SPACES.
05 FCB-SU      PIC X(04)  VALUE SPACES.
05 FCB-DN      PIC X(08)  VALUE SPACES.
05 FCB-SINKID  PIC X(08)  VALUE SPACES.
05 FILLER      PIC X(28)  VALUE SPACES.
05 FCB-ECHO    PIC X(04)  VALUE "ECHO".
05 FILLER      PIC X(20)  VALUE SPACES.
05 FCB-STATUS  PIC S9(5)  COMP-3 VALUE +0.
05 FILLER      PIC X(104) VALUE SPACES.
```

The FCB would be set up as follows:

```
105-SET-UP-FCB-HLI-SU-CMS.
      MOVE "CAR"      TO FCB-FN.
      MOVE "FOCUS"    TO FCB-FT.
      MOVE "A D"      TO FCB-FM.
      MOVE "SULO"     TO FCB-SU.
      MOVE "SINKMA"   TO FCB-SINKID.
```

4. Generate the HLI load module, issuing the GENHLI command using the BOTH option.

**Note:**

- In CMS, all HLIPRINT commands processed by the local machines are logged on their originating local user IDs and not on the sink user ID.
- In CMS, all LOCATION segments must have the same filemode and number as the host file. For example, if the host file has a filemode of A6, the LOCATION files must also have a filemode of A6.

---

## Special Considerations

While you are reading the database with this facility, you won't know if someone else is changing it with a MODIFY or MAINTAIN request. Since any report you generate will be affected by another user's changes, keep the following in mind:

- You can find out if another user is using the database by issuing the ? SU query. If you are the only user on the system when your report is generated, the data will not change.
- If another user is modifying records that your TABLE request will read, your report may contain any combination of modified and unmodified data.

If a root segment instance (or root segment of an alternate view) is deleted and re-inserted while your report is in progress, your report may show the segment instance appearing twice.

---

## CHAPTER 3

# SU and the FOCUS Language

### Topics:

- Using Centrally Controlled Databases
- Messages From the Sink Machine
- MODIFY and MAINTAIN Facility Considerations
- Protecting Sink Transactions
- Performance Considerations
- Using Commands in the SU Profile for COMMIT and ROLLBACK

This chapter discusses reading and modifying centrally controlled databases using the FOCUS language. The chapter is addressed to end users and those who design FOCUS requests.

All FOCUS commands can be used in SU with the following restrictions:

- The SCAN and FSCAN facilities are read-only.
- If you combine databases with the COMBINE command, all databases must be controlled by the same sink machine.
- End users cannot use the REBUILD utility or the RESTRICT command, or issue the command for a database controlled by the Sink Machine.

---

# Using Centrally Controlled Databases

The normal limitations of the USE command for SU apply. You cannot concatenate multiple files, rename files (using the AS keyword), or use the READ parameter for read-only access.

To gain access to centrally controlled databases, specify the databases in an extended form of the FOCUS USE command.

```
USE
fileid1 ON sinkid
fileid2 ON sinkid
.
.
END
```

where:

`fileid1 ...`

Are the database file IDs: filename, filetype, and filemode. Database files under SU must have FOCUS as the filetype. The filemode is defined relative to the sink machine and is usually A.

`sinkid`

Is the user ID of the sink machine. The database must reside on a disk to which the sink machine has write access.

In the USE command, you must specify all the databases you will use under SU, but you can also specify locally controlled databases. For example, if you want to access the CAR file residing on the A disk of the sink machine MULTID, the EMPLOYEE file residing on the A disk of the sink machine SUID, and the PRODUCT file residing on your own J disk, enter:

```
USE
CAR FOCUS A ON MULTID
EMPLOYEE FOCUS A ON SUID
PRODUCT FOCUS J
END
```

Notice in the above example that several sink machines can be active at the same time.

As of FOCUS Release 6.8, ON may be used with AS in the USE command, if the keyword READ is included. The syntax is:

```
Use {ADD      } {fileid} {READ      }
    {CLEAR   }           {NEW       }
    {REPLACE }           {AS mastername}
                                {ON userid }
                                {AS mastername ON userid READ}
                                {LOCAL   }
```

Unless you are using multi-threaded processing, you do not have to link to the disk containing the centrally controlled databases (CMS LINK and ACCESS commands) to access the FOCUS files. However, both the sink and the source machines must be able to read identical Master File Descriptions. If the descriptions are on the source machine user ID, the sink machine user ID should link and gain access to the disk containing the descriptions before the sink machine is started. Alternatively, both the sink and all source machines can link to a separate disk containing the descriptions.

For example, suppose the sink machine is reading Master File Descriptions on user ID MULTID. Then you, too, should gain read access to user ID MULTID to read the same descriptions as the sink machine. For example, enter:

```
CP LINK MULTID 191 200 RR
ACCESS 200 D
```

At this point, you and the sink machine are reading the same Master Files.

You can also use the LOCAL keyword to access sink databases directly through the operating system for reporting. Before issuing the USE command, link and access the minidisk containing the database in read-only mode. The USE command must specify the sink database with the ON option, as shown below, followed by a second entry for the database using the LOCAL option. The filemode for the LOCAL option is the mode in which you accessed the disk. For example:

```
CMS CP LINK MULTID 191 391 RR
CMS ACCESS 391 B
USE
EMPLOYEE FOCUS A ON MULTID
EMPLOYEE FOCUS B LOCAL
END
```

## Messages From the Sink Machine

This section discusses messages from the sink machine. It covers:

- Error messages.
- The ? SU query.

### Error Messages

If the sink machine encounters a problem in processing your FOCUS request, it sends you an error message. The most common problem is that the sink machine is not active. This can happen because:

- You did not specify the correct sink user ID in your USE command.
- The sink machine was never started.
- The sink machine was shut down.

---

When you send a request or transaction to a sink machine that is not operating, you receive one of these error messages:

- FOC542, if you did not specify the correct sink user ID in your USE command or if the sink machine was never started.
- FOC543, if the sink machine was not given enough memory when it was started.
- FOC544, if the sink machine was shut down.

If the sink machine is shut down normally while you are in the middle of executing a MODIFY or MAINTAIN request, all your transactions are written to the database. If the sink machine abends (for example, someone entered HX to shut down the sink machine), some of your last transactions may be lost (to minimize these losses, see *Testing for Rejected Transactions* on page 3-8).

If you use an alternate Master File Description via the sink and the READ keyword is not included in the USE list, the following error message is issued:

(FOC896) READ OPTION REQUIRED ON USE STATEMENT WITH AS AND ON OPTIONS

**Note:** A complete list of error messages appears in Appendix B, *FOCUS Error Messages*.

## Listing Users on the Sink Machine: The ? SU Query

You can display a list of the active source machine user IDs and the file IDs of the sink machine databases from your source machine. The syntax is:

```
? SU [userid]
```

or, for IUCV:

```
? SERVER [userid]
```

where:

`userid`

Is the user ID of the sink machine.

**Note:** For more information on IUCV, see Chapter 5, *Inter User Communication Vehicle (Version 7.0.8 and Up)*.

Issuing the command

? SU

may produce the following:

USERID	FILEID	FOCUS	QUEUE
MKTMN	APPLS	FOCUS	A1
ISDLMR	APPLS	FOCUS	A1
PGMERK	APPLS	FOCUS	A1
HOTCXS	APPLS	FOCUS	A1
CSSVDC	APPLS	FOCUS	A1
MKTDMH	APPLS	FOCUS	A1
MKTMER	APPLS	FOCUS	A1
MKTMRP	APPLS	FOCUS	A1
MKTKAM	APPLS	FOCUS	A1
PSCLM	APPLS	FOCUS	A1
MKTWPF	APPLS	FOCUS	A1

>

This is a convenient facility for monitoring user activities. For example, to inquire about the sink machine MULTID, you enter:

? SU MULTID

Your terminal displays

USERID	FILEID	FOCUS	QUEUE
JBP	CAR	FOCUS	A Q

showing that user ID JBP is now accessing file CAR FOCUS on the A disk of MULTID. The Q in the QUEUE column indicates that user ID JBP is awaiting requests and transactions to be processed.

If no one is using centrally controlled databases on MULTID, the following message appears:

NO SIMULTANEOUS USERS ARE ACTIVE ON MULTID

If the sink machine is not operating on MULTID, error message FOC542, FOC543, or FOC544 will appear, as described in *Error Messages* on page 3-3.

---

## Testing the Status of ? SU Query

The ? SU query may be used to provide test values in FOCEXEC procedures. When ? SU is executed, the query results determine the status return variable of Dialogue Manager (RETCODE). Query results and their corresponding values are:

Query Result	RETCODE Value
Users are active on sink machine.	0
No users are active and sink machine is running.	8
Sink machine is not running.	16

The RETCODE value is later tested in an IF statement. For example, this procedure tests whether the sink machine is available before starting your MODIFY or MAINTAIN procedure

```
? SU sinkid
-RUN
-IF &RETCODE EQ 16 GOTO BAD;
-INCLUDE myproc
-BAD
-EXIT
```

where:

`sinkid`

Is the user ID of the sink machine.

`myproc`

Is the name of your MODIFY or MAINTAIN procedure.

If the value of RETCODE is 0 or 8, your specified MODIFY or MAINTAIN procedure will be executed.

## MODIFY and MAINTAIN Facility Considerations

This section discusses considerations that you should make when using MODIFY or MAINTAIN to modify centrally controlled databases. This section covers:

- The FOCURRENT variable, which indicates if your transaction was rejected because of a conflict.
- Using the FOCURRENT variable to test for conflicts.
- The FOCURRENT variable and CRTFORM turnaround fields.

## Indicating Conflicts: The FOCURRENT Variable

When you submit a MODIFY or MAINTAIN transaction in SU, FOCUS stores a code in a variable called FOCURRENT. This variable indicates whether or not there is a conflict with another transaction as determined by change verify protocol (explained in Chapter 1, *Introduction*). If the variable value is 0, there is no conflict and the transaction is accepted. If the value is not 0, there is a conflict. In this case, the transaction is rejected and an error message appears

FOCUS treats a MODIFY or MAINTAIN transaction that has been rejected because of a conflict as if it failed a validation test (the VALIDATE statement). You can log these transactions using the LOG INVALID statement described in Chapter 15 of the *FOCUS Users Manual*.

You can also design your MODIFY or MAINTAIN requests to test FOCURRENT and branch according to its value. For example, a MODIFY or MAINTAIN request can submit a transaction, test FOCURRENT, and, if FOCURRENT is a non-0 value, resubmit the transaction. This is explained in *Testing for Rejected Transactions* on page 3-8.

The values assigned to FOCURRENT are:

- 0 Accepted.
- 1 Invalid, input will create duplicate.
- 2 Invalid, instance now deleted.
- 3 Invalid, instance has been changed.

The following table shows the possible values of FOCURRENT after different types of transactions. The rows list each type of transaction; the columns list the possible results. A hyphen (-) indicates that the transaction is rejected for reasons other than change verify protocol.

User-Desired Action	No Simultaneous Action	Instance Simultaneously:		
		Updated	Deleted	Included
UPDATE	0	3	2	-
DELETE	0	3	2	-
INCLUDE	0	-	-	1

In summary, the FOCURRENT variable is not 0 if:

- You cannot update or delete a segment instance, because another user has changed or deleted the record.
- You cannot include a new instance, because another user has added the same instance.

---

There is one additional case in which FOCURRENT is set to a non-zero value. If you issue a MATCH for a particular set of key values, the sink machine saves the values of the retrieved segment instance. If you rematch on the same key values without intervening MATCH or NEXT commands against the database, the sink machine compares the newly retrieved segment instance with the copy it saved on the first MATCH. If these two segment instances are the same, FOCURRENT is set to 0. If the segment instances differ, another user must have changed the segment instance since the original MATCH was performed. In this case, FOCURRENT is set to 1. This is the only case where FOCURRENT is set to a non-zero value on a MATCH command.

An example of this particular situation is given in *Validating CRTFORM Turnaround Fields* on page 3-9.

## Testing for Rejected Transactions

By testing the FOCURRENT variable, MODIFY or MAINTAIN requests can process transactions even after they have been rejected because of conflicts. You design these requests using Case Logic (Case Logic is discussed in Chapter 15 of the *FOCUS Users Manual*).

For instance, assume a MODIFY or MAINTAIN request submits a transaction that is rejected. It then repeats the transaction. In this case, there are two possible results:

- If the transaction was previously rejected because the database instance was already updated, the transaction may be accepted the second time.
- If the database instance was deleted, the request itself will reject the transaction through MATCH/NOMATCH logic.

In general, it is safe to branch to the same case again, as deadlocks cannot occur between two users. A transaction that resulted in a MATCH condition may, on resubmission, result in a NOMATCH condition, as the following request illustrates:

```
MODIFY FILE EMPLOYEE
PROMPT EMP_ID
GOTO NEWSAL

CASE NEWSAL
MATCH EMP_ID
    ON NOMATCH REJECT
    ON MATCH PROMPT CURR_SAL
    ON MATCH UPDATE CURR_SAL
    ON MATCH IF FOCURRENT NE 0 GOTO NEWSAL;
ENDCASE
DATA
```

The request prompts you for an employee ID and branches to the NEWSAL case. If the ID is in the database, the NEWSAL case prompts you for a salary, updates the salary on the source machine copy of the instance, and submits the transaction.

The procedure then tests the value of the variable FOCURRENT. If FOCURRENT is 0, the transaction was accepted and the request prompts you for the next ID. If FOCURRENT is not 0, meaning that the transaction was rejected, the request branches back to the top of the NEWSAL case and searches again for the employee ID in the database.

If the instance with the employee ID is still there, it prompts you again for the salary and resubmits the transaction. But if the instance was deleted, the request reports back a NOMATCH condition (ON NOMATCH REJECT) and prompts you for the next transaction.

## Validating CRTFORM Turnaround Fields

You should include FOCURRENT tests in MODIFY and MAINTAIN requests that validate CRTFORM turnaround fields. Otherwise, after a transaction is rejected as invalid, you may update a field without knowing that the field has been updated by someone else.

Turnaround fields are fields with names having the T. prefix that appear in CRTFORMs in MODIFY and MAINTAIN update requests (for example, T.SALARY). When you execute such a request, the present database values of the turnaround fields are displayed in the CRTFORM where you can change them or accept the values shown. (For more information on turnaround fields, see Chapter 16 in the *FOCUS Users Manual*.)

You can subject turnaround fields to validation tests. If you enter an invalid value for a field, the request rejects the value, retrieves a fresh copy of the segment instance from the database, but redisplay the turnaround field values on the screen as you entered them (after you press ENTER a second time).

Since the request redisplay the values from your first attempt, you do not know if another user updated these values after you began the transaction. Change verify protocol will not reject the transaction, because the request retrieved a fresh copy of the instance after your entries failed the validation tests (this copy contains the other user's updated values, but these values are not displayed).

Therefore, your MODIFY or MAINTAIN request should test the FOCURRENT variable after you retrieve data from the database with the MATCH command. If you have previously MATCHed on the same key values, and the instance currently in the database is not the same as your version of the instance, FOCURRENT is set to the value of 1.

The following example shows why the FOCURRENT test is needed and how it should be used.

---

This request updates employees' salaries, allowing each employee a maximum salary of \$50,000 yearly:

```
MODIFY FILE EMPLOYEE
CRTFORM
    "ENTER EMPLOYEE'S ID: <EMP_ID"

MATCH EMP_ID
    ON NOMATCH REJECT
    ON MATCH CRTFORM LINE 2
        "ENTER SALARY: <T.CURR_SAL"
    ON MATCH VALIDATE
        SALTEST = IF CURR_SAL LE 50000 THEN 1 ELSE 0;
    ON INVALID TYPE
        "INVALID SALARY: PLEASE CORRECT VALUE"
    ON MATCH UPDATE CURR_SAL
    ON MATCH IF FOCURRENT NE 0 GOTO ERROR;

CASE ERROR
TYPE
    "FOCURRENT TEST FAILED"
    "ANOTHER USER HAS CHANGED THIS SEGMENT INSTANCE"
GOTO TOP
ENDCASE
DATA VIA FIDEL
END
```

The request prompts you for the employee ID number. After you enter the ID 071382660, FOCUS displays:

```
ENTER EMPLOYEE'S ID:      071382660
ENTER SALARY:             11000.00
```

Unknown to you, another user now updates the same instance, giving the employee a salary of \$13,000. You change the salary to \$15,000:

```
ENTER EMPLOYEE'S ID:      071382660
ENTER SALARY:             55000.00
```

But you see that you have made a mistake: you entered a salary of \$55,000, which exceeds the \$50,000 maximum. You receive a message that you entered an invalid salary value. You press ENTER to redispatch the turnaround fields. FOCUS responds:

```
ENTER EMPLOYEE'S ID:      071382660
ENTER SALARY:             55000.00
```

Notice that the SALARY field still displays 55000.00, not the updated value 13,000, even though the request has just retrieved a fresh copy of the instance from the database. You change the salary to \$15,000:

```
ENTER EMPLOYEE'S ID:      071382660
ENTER SALARY:             15000.00
```

When you press ENTER, these values are entered into the database, canceling the other user's update (which may be correct).

To solve the problem, have the request test the FOCURRENT variable when it retrieves copies of instances from the database. The previous MODIFY request could be written in the following manner (addition is underlined>:

```

MODIFY FILE EMPLOYEE
CRTFORM
  "ENTER EMPLOYEE'S ID: <EMP_ID"

MATCH EMP_ID
  ON NOMATCH REJECT
  ON MATCH IF FOCURRENT NE 0 GOTO ERROR;
  ON MATCH CRTFORM LINE 2
    "ENTER SALARY: <T.CURR_SAL"
  ON MATCH VALIDATE
    SALTEST = IF CURR_SAL LE 50000 THEN 1 ELSE 0;
ON INVALID TYPE
  "INVALID SALARY: PLEASE CORRECT VALUE"
ON MATCH UPDATE CURR_SAL
ON MATCH IF FOCURRENT NE 0 GOTO ERROR;

CASE ERROR
TYPE
  "FOCURRENT TEST FAILED"
  "ANOTHER USER HAS CHANGED THIS SEGMENT INSTANCE"
GOTO TOP
ENDCASE
DATA VIA FIDEL
END

```

Now when you enter the EMP\_ID key, this MODIFY request first retrieves a copy of the instance with that EMP\_ID value, then tests the FOCURRENT value. If FOCURRENT is not 0, the request branches back to the beginning and you must enter the EMP\_ID number again. This will display the current turnaround field values in the database.

If you execute the previous request and enter an invalid salary, FOCUS clears the turnaround fields and notifies you of the failed validation test as before. But when you press ENTER again, FOCUS tests the FOCURRENT variable. If another user has changed the value of SALARY since the first time you retrieved it, the reMATCH on EMP\_ID will return a FOCURRENT value of 1. If FOCURRENT is not 0, CASE ERROR informs you that your transaction has failed the FOCURRENT test and requests that you reenter the EMP\_ID value. Only when you enter the EMP\_ID value again does FOCUS retrieve a fresh copy of the instance, displaying:

```

ENTER EMPLOYEE'S ID:      071382660
ENTER SALARY:             13000.00

```

FOCUS displays the SALARY value as 13000.00, which is the updated value. You can now decide to leave this value or change it.

---

# Protecting Sink Transactions

MODIFY and MAINTAIN each provide two ways of protecting transactions in case a system failure brings down the sink machine:

- The Checkpoint facility.
- The subcommands COMMIT and ROLLBACK.

A discussion of each method follows.

## The Checkpoint Facility

When FOCUS accepts transactions, it does not write the transactions to the database immediately; rather, it collects them in a buffer. When the buffer is full, FOCUS writes them all to the database at the same time. This cuts down on the number of input/output operations that FOCUS must perform. However, if your CMS system fails and brings down the sink machine, the transactions collected in the buffer may be lost. You can cause FOCUS to write more frequently to the database by using the Checkpoint facility, described in Chapter 15 of the *FOCUS Users Manual*.

The Checkpoint facility is activated with the MODIFY CHECK statement that specifies how many transactions accumulate in the buffer. The request below writes transactions to the database in groups of 10:

```
MODIFY FILE EMPLOYEE
PROMPT EMP_ID CURR_SAL
MATCH EMP_ID
    ON NOMATCH REJECT
    ON MATCH UPDATE CURR_SAL
CHECK 10
DATA
```

Under SU, when FOCUS writes transactions for a single user on a sink machine, it automatically writes the transactions of all users on that sink machine even for those users who did not request a checkpoint action. It then records this as a SAV action in the HLIPRINT file. FOCUS also writes the transactions of all users on the same sink machine when one user's MODIFY or MAINTAIN request finishes execution. This is recorded as a CLO action in the HLIPRINT file.

If the system fails while MODIFY or MAINTAIN requests are executing, all transactions entered before FOCUS last wrote to the database (because of the Checkpoint facility or because a MODIFY or MAINTAIN request finished execution) are saved in the database. All transactions entered after that may be lost and must be reentered. Use CHECK 1 to avoid this situation.

It is sometimes a good idea to log all transactions in a sequential file so that they may be reentered if any are lost. To do this, add LOG or TYPE statements to the MODIFY or MAINTAIN request. LOG statements are satisfactory for simple requests; TYPE statements are better for complex Case Logic requests. For more information, see Chapter 15 in the *FOCUS Users Manual*.

## Managing MODIFY and MAINTAIN Transactions: COMMIT and ROLLBACK

COMMIT and ROLLBACK are two subcommands. COMMIT gives you control over the content of database changes and ROLLBACK enables you to undo changes before they become permanent.

The COMMIT subcommand safeguards transactions in case of a system failure and provides greater control (than the Checkpoint facility) over which transactions are written to the database.

The MODIFY (or MAINTAIN) CHECK statement only enables you to control the number of transactions that must occur before changes are written to the database. When using CHECK, you cannot change the Checkpoint setting once the request begins execution. Similarly, changes cannot be canceled (see *The Checkpoint Facility* on page 3-12 for information on the CHECK statement).

COMMIT enables you to make changes based on the content of the transactions as well as the number. Changes you do not want to make can be canceled with ROLLBACK, unless a COMMIT has been issued for those changes. Should the system fail, either all or none of your transactions will be processed.

Absolute File Integrity is required in order to use COMMIT and ROLLBACK. This is automatic for all databases that do not have filemode 6 (for example, CAR FOCUS A6). See Chapter 21 of the *FOCUS Users Manual* for information on Absolute File Integrity and the SET SHADOW command.

### COMMIT and ROLLBACK Subcommands

COMMIT and ROLLBACK each process a logical transaction. A logical transaction is a group of database changes in the MODIFY or MAINTAIN environment that you want to treat as one. For example, you can handle multiple records displayed on a CRTFORM and then processed via the REPEAT command as a single transaction. A logical transaction is terminated by either COMMIT or ROLLBACK. COMMIT and ROLLBACK also can be used for single-record processing.

---

When COMMIT ends a logical transaction, it writes all changes to the database. COMMIT can be coded as a global subcommand or as part of MATCH or NEXT logic. The possible MATCH and NEXT statements are:

```
COMMIT
ON MATCH COMMIT
ON NOMATCH COMMIT
ON MATCH/NOMATCH COMMIT
ON NEXT COMMIT
ON NONEXT COMMIT
```

When ROLLBACK ends a logical transaction, it does not write changes to the database. The ROLLBACK subcommand cancels changes made since the last COMMIT. ROLLBACK cannot cancel changes once a COMMIT has been issued for them.

ROLLBACK can be coded as a global subcommand or as part of MATCH or NEXT logic. Possible MATCH and NEXT statements are:

```
ROLLBACK
ON MATCH ROLLBACK
ON NOMATCH ROLLBACK
ON MATCH/NOMATCH ROLLBACK
ON NEXT ROLLBACK
ON NONEXT ROLLBACK
```

If the COMMIT fails for any reason (for example, system failure, lack of disk space), no changes are made to the database. In this way, COMMIT is an all-or-nothing feature that ensures database integrity.

## Using COMMIT and ROLLBACK With the Sink Machine

The change verify protocol in SU operates on a logical transaction basis that may encompass a block of records. It relies on the “optimistic” assumption that two users rarely change the same records at the same time. In SU processing with COMMIT, the sink keeps a table of soft locks that tracks which records have been requested by which users.

When a COMMIT is issued, the sink checks this table for each record you want to change to determine if another user changed a record between the time you retrieved it and the time you issued the COMMIT. If another user has issued a COMMIT against a record you intend to change, your changes are automatically rolled back.

For example, User A and User B execute a MODIFY or MAINTAIN procedure against the EMPLOYEE database on a sink machine. Each user matches on the same employee ID and performs a MATCH action as follows:

User A	User B
1. Matches on employee ID 123.	1. Matches on Employee ID 123.
2. Updates current salary to \$50,000.	2. Deletes employee 123.
3. COMMIT	3. COMMIT

If User A issues a COMMIT before User B, User B's changes will be rolled back. If User B issues a COMMIT before User A, User A's changes will be rolled back.

Many applications contain transactions that are interdependent. For these applications, processing is successful only when several physical transactions are complete. The following application transfers funds from one bank account to another. Two components make up each logical transaction:

- The subtraction of an amount of money from one account.
- The addition of the same amount to another account.

If either part of the logical transaction is not complete, no changes should be made to the database. Because the application has multiple users, it is necessary for each account to be involved in only one transfer transaction at a time.

This application uses the BANK file, a FOCUS database on a sink machine with the following Master File Description:

```
FILE = BANK, SUFFIX = FOC,$
SEGNAME = TOPSEG, SEGTYPE = S1,$
  FIELD = ACCOUNT_NUM      , ACCT      , A8      , $
  FIELD = BALANCE          , BAL      , D12.2    , $
  FIELD = CUSTOMER        , NAME     , A40     , $
```

---

The following MODIFY procedure shows how COMMIT and ROLLBACK can be used to ensure that a logical transaction is complete before it is written to the database. It also shows the use of FOCURRENT in SU COMMIT processing.

```
MODIFY FILE BANK
COMPUTE AMOUNT = ;
      FROM ACCOUNT/A8 = ; TO ACCOUNT/A8 =
1. CRTFORM
  "FUND TRANSFER. ENTER ACCOUNT NUMBERS:"
  "   TO TRANSFER FROM == > <FROM ACCOUNT"
  "   TO TRANSFER TO   == > <TO ACCOUNT"
  "   ENTER AMOUNT    == > <AMOUNT"
2. PERFORM SUBTRACT
5. PERFORM ADD
7. COMMIT
8. IF FOCURRENT EQ 0 GOTO TOP;
   TYPE "ACCOUNTS WERE IN USE. PLEASE TRY AGAIN."
   GOTO TOP

2. CASE SUBTRACT
  COMPUTE ACCOUNT NUM = FROM ACCOUNT;
  MATCH ACCOUNT NUM
3.   US/AN`ON NOMATCH ROLLBACK
     ON NOMATCH REJECT
     ON MATCH COMPUTE BALANCE = D.BALANCE - AMOUNT;
4.   US/AN`ON MATCH IF BALANCE LT 0 PERFORM REDO;
     ON MATCH UPDATE BALANCE
   ENDCASE

5. CASE ADD
  COMPUTE ACCOUNT NUM = TO ACCOUNT;
  MATCH ACCOUNT NUM
6.   ON NOMATCH ROLLBACK
     ON NOMATCH REJECT
     ON MATCH COMPUTE BALANCE = D.BALANCE + AMOUNT;
     ON MATCH UPDATE BALANCE
   ENDCASE

4. CASE REDO
  TYPE "BALANCE WILL BE LESS THAN ZERO. TRY ANOTHER   AMOUNT."
  GOTO TOP
  ENDCASE

DATA
END
```

1. The CRTFORM requests the account from which to take funds, the account to which funds will be transferred, and the amount to be transferred.
2. The transfer of funds from the first account is performed in CASE SUBTRACT.
3. If the account is not in the database, any changes that may have already been made are rolled back.

4. The new balance is checked before an update takes place. If the balance is less than zero, the user goes to the top case. No UPDATE or COMMIT occurs at this point.
5. The transfer of funds to the second account is performed in CASE ADD.
6. If the account is not in the database, any changes that may have already been made are rolled back. This is particularly important in CASE ADD, because funds may have already been subtracted (the UPDATE in CASE SUBTRACT).
7. The COMMIT is executed if both updates are successful.
8. A test for a zero value of FOCURRENT is placed after the COMMIT statement. If the COMMIT fails (FOCURRENT has a non-zero value) the transaction will be rolled back.

**Note:**

- COMMIT and ROLLBACK are available only for FOCUS files; they will be ignored in MODIFY and MAINTAIN procedures against non-FOCUS files.
- All files referenced by a MODIFY or MAINTAIN procedure that use COMMIT and ROLLBACK processing must be specified in a USE statement. This includes cross-referenced files and joined files.
- After a COMMIT or ROLLBACK command against a sink database, the procedure has no current position in the file. You must reestablish position with a MATCH or NEXT statement.
- COMMIT and ROLLBACK functionality is not available for HLI.
- COMMIT and ROLLBACK do not support use of an alternate view. For example, MODIFY filename.field cannot be used.
- COMMIT and ROLLBACK cannot be used to process S0 segments.
- Text fields cannot be used in a MODIFY or MAINTAIN that uses COMMIT and ROLLBACK.
- The maximum size of the buffer used for COMMIT transactions to be sent to the sink machine is 32K bytes.

---

## Performance Considerations

COMMIT and ROLLBACK improve SU performance because the ability to group individual transactions as one logical transaction requires fewer disk I/Os. Reducing the number of individual transactions also reduces the number of communications between the sink machine and source user IDs. Specifically, only one communication takes place when transactions are rolled back, and no communication occurs when a record is matched a second time. Both of these situations are common with multi-record processing.

## Comparing COMMIT and ROLLBACK to the Checkpoint Facility

When using CHECK processing, the sink machine buffers all changes to BINS before saving your changes to the database. The changes are saved from BINS to disk when a checkpoint is issued via the CHECK command in MODIFY (or MAINTAIN). For example, CHECK 5 tells the MODIFY or MAINTAIN procedure to issue a checkpoint after every five changes to the database.

The CHECK facility has several limitations. There is little control over when checkpoints are done. When one user on a sink machine issues a checkpoint, the sink saves all other users' changes. Also, changes cannot be undone.

The use of COMMIT and ROLLBACK provides a number of advantages over the CHECK facility. All changes are held by the source machine until a COMMIT is issued, and then the changes are sent to the sink. No one else sees your changes until you issue a COMMIT. The changes can be canceled by the ROLLBACK command.

COMMIT and ROLLBACK are controlled by the MODIFY or MAINTAIN procedure and are not subject to the checkpoints of other users. All or nothing processing is guaranteed, even upon system failure. Because all the changes are sent to the sink machine in one message, processing is more efficient.

## Using FOCURRENT

FOCURRENT logic is necessary for determining whether or not transactions have been accepted. In an application without COMMIT and ROLLBACK, FOCURRENT cannot deal with multiple changes and may be difficult to code.

When COMMIT and ROLLBACK are used, FOCURRENT should be tested after each COMMIT. If the COMMIT fails, the value of FOCURRENT is set to a non-zero value. MODIFY or MAINTAIN only checks FOCURRENT after a COMMIT, not after every transaction. If the value of FOCURRENT is not zero, MODIFY or MAINTAIN knows that the entire logical transaction was rolled back.

Note also that if FOCURRENT has a non-zero value, no message is issued. Therefore, you should test FOCURRENT explicitly after a COMMIT and take any appropriate action; otherwise no message is displayed and processing returns to the top case.

## Seeing Other Users' Changes While Using COMMIT and ROLLBACK

Changes by other users can be seen only after a COMMIT has been issued and processed successfully. If you MATCH or NEXT on a record more than once and you have not issued a COMMIT, the MODIFY or MAINTAIN does not go to the sink again. This means that you will not see other users' changes until your MODIFY or MAINTAIN executes a COMMIT statement, although you will see your own changes. Once a COMMIT is performed, the record will come from the database if you MATCH or NEXT on it.

This concept is especially important when the MODIFY or MAINTAIN uses FIND or LOOKUP. These operations cause records to be retrieved from the sink database. FIND and LOOKUP do not search for values on the local copies of the records. For example, if you have included a record and then use LOOKUP to obtain a value on this record, the value will not be found if you have not issued a COMMIT.

## Using Commands in the SU Profile for COMMIT and ROLLBACK

Two commands are available for use in the SU profile that enable the sink database administrator to control whether COMMIT processing is allowed and how the sink checks for currency errors. (The SU Profile is discussed in Chapter 2, *Operating the Sink Machine*.)

## Controlling COMMIT Processing on the Sink Machine

The sink machine, by default, accepts source users whose MODIFY or MAINTAIN procedures include the COMMIT and ROLLBACK subcommands.

---

## Controlling Currency Checking on the Sink Machine

The sink machine by default determines the value of FOCCURRENT by checking the segment that was updated and all the parent segments in the path. If any segment in a path has been changed between the time a source user retrieved the record and the time the user issues a COMMIT, FOCCURRENT will have a non-zero value and the COMMIT will fail.

This default can be changed for the sink with the SET PATHCHECK command so that only the updated segment is checked. Syntax for the SET PATHCHECK command, which is placed in the SU profile, is

```
SET PATHCHECK = {ON }  
               {OFF}
```

where:

**ON**

Is the default. Checks parent segments as well as the updated segment to determine the value of FOCCURRENT. This will occur for all source users.

**OFF**

Checks only the updated segment to determine the value of FOCCURRENT. Applies to all source users.

---

## CHAPTER 4

# SU and the Host Language Interface (HLI)

### Topics:

- Creating an HLI Module
- The File Communication Block (FCB)
- Closing Centrally Controlled Databases
- Writing Transactions From the Buffer
- Stopping a Sink Machine Using HLI Control Commands
- Communications Protocol

This chapter describes how to use the FOCUS Host Language Interface (HLI) to read and modify centrally controlled databases. (The FOCUS Host Language Interface enables programs written in COBOL, FORTRAN, PL/1, or Assembler to read and modify data in FOCUS files.) It describes how two or more users can use HLI in SU to read and modify the same database concurrently. Each user is unaware of other users and may retrieve, add, delete, or change data independently.

All programs address the same copy of HLI, which is executed by the sink machine. When a program on a source machine issues a FOCUS call containing an HLI command, the FOCUS subroutine transmits the call to the copy of HLI on the sink machine, which then executes the command. You do not need to change programs designed to run in local mode; however, you must insert certain parameters into the File Communication Block (see *The File Communication Block (FCB)* on page 4-2).

This chapter assumes that you are already familiar with HLI.

---

## Creating an HLI Module

To use centrally controlled databases with HLI, you must combine your program and subroutines together with HLI routines into a single module. To do this, first designate the text libraries you need to run your program using the CMS GLOBAL command. Then enter

```
FOCUS GENHLI text1 text2 ... textn (SU
```

where:

```
text1...
```

Are the names of your programs and subroutines in text form that you want to combine with HLI routines in the module. There is no closing parenthesis after “SU”.

This command executes a CMS EXEC procedure that generates the module on your A disk. The module has the same filename as the first routine you specified (that is, text1 in the syntax above). For example, the command

```
FOCUS GENHLI GRADE PRIME SUBROUT1 (SU
```

combines the text routines GRADE, PRIME, and SUBROUT1 with HLI routines into the single module GRADE MODULE A1, which can run in CMS mode. To run this module, enter GRADE from the CMS command level.

If you need a module that can process both centrally controlled databases and local FOCUS files, use the option BOTH at the end:

```
FOCUS GENHLI text1 text2 ... textn (BOTH
```

## The File Communication Block (FCB)

All HLI commands start with the command name as the first argument in the call, followed by the File Communication Block (FCB) as the second argument. For example:

```
CALL FOCUS ('OPN ',MYFCB)
```

The FCB is always 200 bytes with a fixed layout. The user program supplies some of the parameters in the FCB, such as the CMS fileid (filename, filetype, and filemode) of the database. HLI fills in the other parameters, such as the status return code.

When running a program under SU, place the value SU b/ b/ in FCB word 6. (**Note:** Two blanks must follow the characters “SU”.) Place the user ID of the virtual machine that will process the HLI calls (the sink machine) in FCB words 9 and 10. These are the only changes you must make to your HLI programs to have them run under SU.

## FCB Area Layout

The layout of the FCB area is shown below.

FCB Words	Item	Meaning	Bytes Used
1 - 2	FN	CMS and FOCUS filename of database (or HLI control command)	8
3 - 4	FT	CMS filetype of database	8
5	FM	CMS filemode of database	4
6	SU	Equal to value SU b/ b/ or blank	4
7 - 8	PROCNAME	Displayed on HLI PRINT	8
9 - 10	SINKID	Userid of sink machine	8
11 - 12	-	Reserved	8
13 - 15	-	Reserved	12
16 - 17	BACKKEY	Address key of target segment	8
18	ECHO	ECHO or STAT	4
19 - 20	PASSCTL	File access password	8
21 - 22	NEWSEG	Name of highest new segment retrieved	8
23	SEGNUM	Segment number of NEWSEG (integer)	4
24	STATUS	Status return code (integer)	4
25	ERRORNUM	Detail error code (integer)	4
26 - 32	-	Reserved	28
33	SHOLENGTH	Length of one record returned	4
34	LENGTH	Length of work area returned	4
35 - 50	-	Reserved	64

Note: FCB words 5 and 6 each must be padded with two trailing blanks.

**Figure 4-1. Layout of the File Communication Block**

Note that the HLI program can place information useful for debugging in words 7 and 8 of the FCB. This information is displayed on the HLI PRINT trace in the PROCNAME column.

---

## Using Both Local and Centrally Controlled Databases

A single program may use a combination of centrally controlled databases and local FOCUS files. Whenever the program is using the centrally controlled databases, the FCB must have the characters SU\_\_ in word 6 and the sink machine user ID in words 9 and 10. Under HLI, each open file must have its own FCB.

## Error Codes

After every HLI call is executed, a status code is placed in the File Communication Block (FCB) word 24, shared by the host program and HLI. A status code of 0 means that the command executed correctly. A status code of 1 means that the command executed correctly, but no more data of the type requested was available. All higher-numbered status codes indicate error conditions such as incorrect fieldnames or missing parameters. A list of status codes is provided in Appendix C, *HLI Status Codes Returned in FCB Word 24*.

If you are modifying the database using the INP, CHA, and DEL commands and another user is trying to modify the same segment instance at the same time, your transaction may be rejected because the other user's transaction was accepted first (this process is explained in Chapter 1, *Introduction*). If this happens, your FCB receives one of the following status codes:

### 784

You attempted to add a new segment instance. Your transaction was rejected, because another user added the same segment instance first. (Analogous to FOCURRENT=1 in MODIFY/SU).

### 786

You attempted to update or delete a segment instance. Your transaction was rejected, because another user deleted the instance first. (Analogous to FOCURRENT=2 in MODIFY/SU).

### 789

You attempted to update or delete a segment instance. Your transaction was rejected, because another user updated the instance first. (Analogous to FOCURRENT=3 in MODIFY/SU).

If word 24 is a non-0 value, FCB word 25 may contain the detail error code. This word 25 indicates a VMCF or IUCV error, which is a subcategory of the error specified by the status code in FCB word 24. If word 25 is not 0, record the code and contact your IBI representative.

**Note:** Status code 802, detail error code 5, indicates the sink machine terminated abnormally.

## Closing Centrally Controlled Databases

Before finishing execution, your program should issue the CLO command to close the FOCUS databases it opened. If your program does not, HLI closes the databases for you, even if your program abends.

If another user closes a file you are also using, the sink machine saves both that user's changes and your changes made up to that point.

## Writing Transactions From the Buffer

When users make changes to the database (the CHA command), HLI stores these changes in buffer or a shadow page until one of the following happens:

- A program finishes execution.
- A program closes down a FOCUS file when CLO is issued.
- A program issues a SAV command.

When one of these occurs, HLI writes all the users' changes to the database.

## Stopping a Sink Machine Using HLI Control Commands

HLI control commands can shut down the sink machine from one of the source machines. The syntax for control commands is

```
CALL FOCUS ('HLI ', fcb)
```

where:

`fcb`

Is the File Communication Block.

Prepare the FCB as follows:

- Place the subcommand to be executed in the first 8 bytes of the FCB (replaces filename in FCB words 1 and 2). These subcommands are:

`STOPSINK`

Causes the sink machine to execute any pending requests in queue, to close all files, and to return to CMS.

`HX`

Terminates the sink machine abnormally and returns to CMS. Use only in an emergency, because it may leave files vulnerable to damage.

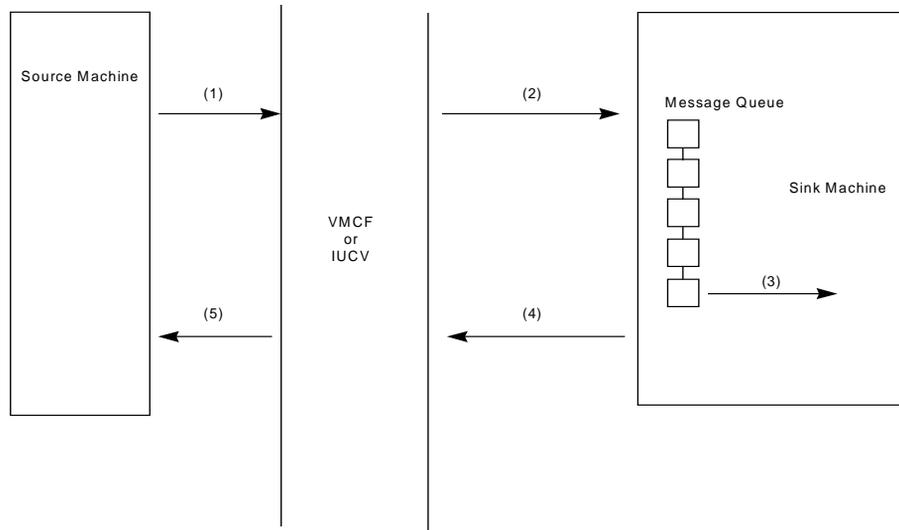
- Place the value SUB/ b/ in FCB word 6 with two trailing blanks.

- Place the user ID of the sink machine you intend to shut down in FCB words 9 and 10.
- Place the control password, entered at the time the sink machine was started (described in Chapter 2, *Operating the Sink Machine*), in FCB words 19 and 20.

## Communications Protocol

VMCF and IUCV are CMS facilities for transferring data between virtual machines. SU uses the VMCF or IUCV SEND/RECV protocol. The source machine sends HLI commands to the sink machine. The sink machine executes each command and sends the output back to the source machine. If, while executing an HLI command, the sink machine receives another command, it puts the second command on a message queue which it processes in the order received.

Figure 4-2 illustrates this process. A source machine sends an HLI command to VMCF or IUCV (1). The command is transmitted to the sink machine, which puts the command on its message queue to await processing (2). The sink machine accepts the command from the message queue (3), executes the command, and sends the output to VMCF or IUCV (4). VMCF or IUCV transmits the output back to the source machine (5).



**Figure 4-2. VMCF and IUCV Protocol for HLI in the SU Environment**

The time it takes VMCF or IUCV to transfer data back and forth (HLI commands and test literals to the sink machine, field values back to the source machine) depends on the size of your record work area. The larger the work area, the longer it takes VMCF or IUCV to transfer your data. It is a good idea to use the SHO command to limit your record work area to the fields you need so that the work area is as small as possible.

The HLI NEX command has a multiple record option that packs several records into a single-blocked record that VMCF or IUCV transfers to a single record area. VMCF or IUCV can transfer this single-blocked record faster than it can transfer the individual records separately.

---

## CHAPTER 5

# Inter User Communication Vehicle (Release 7.0.8 and Up)

### Topics:

- Starting a Sink Machine Using IUCV
- VM Command Line Support
- Server Commands
- Configuration and Data Files
- CP SMSG Support
- Security Considerations
- EXEC Services
- File Checkpoint
- LOADLIB GLOBAL
- Unsolicited Interrupts
- FILEDEFs
- Sample Directory

The Inter User Communications Vehicle (IUCV) is a communications facility that allows a program running in a virtual machine to communicate with other virtual machines, with a CP system service, and with itself.

An IUCV communication takes place between a source communicator and a target communicator. The communication takes place over a predefined linkage called a path. Each communicator can have multiple paths, and can receive or send multiple messages on the same path simultaneously. This helps avoid programming conflicts by taking advantage of a multi-processing environment.

**Note:** The Inter User Communication Vehicle (IUCV) is only available with FOCUS Release 7.0.8 and higher.

---

## Starting a Sink Machine Using IUCV

New server command line options are as follows. Note that all options default to “NO”. The syntax is:

```
IUCVMAIN password {ECHO|STAT} [options ]
```

where:

*password*

Is a positional password. This is included for compatibility only. Special passwords include ACIGROUP, LOGONPW, NOPASSWD. NOPASSWD may also be used to define no password, but allow the ECHO/STAT parameters, which are also positional.

ECHO

Records activity in a simple HLIPRINT file. NOECHO disables recording. If ECHO or STAT is specified, a default FILEDEF for HLIPRINT with the correct LRECL of 88 or 133 will be issued. If a FILEDEF for HLIPRINT already exists, then this FILEDEF will be honored.

In all cases of FILEDEF conflicts (such as STAT with an LRECL of 88) or a device that does not support the defined LRECL (such as PUNCH with LRECL 133), the LRECL will be corrected.

STAT

Records activity in an extended HLIPRINT file. NOSTAT disables recording.

The following table lists valid options. Note that all options default to “NO”.

Option	Function
VERBOSE	Enables informational messages during the server operation. NOVERBOSE disables informational messages.
DEBUG	Enables a multitude of informational messages. These are issued each time a module is called. NODEBUG disables these messages. Note: VERBOSE and DEBUG are mutually exclusive.
SMSG	Enables the server to accept CP SMSG commands. NOSMSG disables SMSG support.
SNAP	Enables SNAP DUMPS (symptom records) via DIAG x'94' at PROGRAM CHECK or SVC 13 (abend). NOSNAP disables this feature.
REIPL	Issues a CP TRACE command that will IPL CMS at program check. This operation will occur before any program check handler. NOREIPL deletes the CP TRACE set.

Option	Function
MONITOR	Will cut CP MONITOR records, if server is authorized with OPTION APPLMON. If the server is not authorized, it will still monitor server activity, but this will not be sent to CP MONITOR. This can still be helpful for some server problems.
MONNOCUT	Allows a user with APPLMON privileges to start monitor without sending these records to CP.
PASSWORD	With no option, is the same as defining no password. PASSWORD normally takes an option of a server-defined password or a special password. Special passwords include: ACIGROUP, LOGONPW, NOPASSWD. NOPASSWD may also be a separate command line option, which also resets the password to the default of having no password. NOPASSWD resets the password back to the default of no server password.
GTFNSS	With no option, is the same as defining nogtfns trace. GTFNSS normally takes an option of the name of the segment where TRACE records will be cut. NOGTFNSS disables writing TRACE records to the segment.
MFD	The default, indicates the server issues its own CMS file OPEN and CLOSE checkpoints. NOMFD disables this function, which allows CMS to handle file IO in a standard manner.
GLOBAL	The default, indicates the server will issue its own global LOADLIB, replacing any currently defined user globals. NOGLOBAL forces the server to leave the LOADLIB list alone.
GLOBUPDT	Adds the IBI LOADLIBs at the front of the global list, but leaves any globalized user LOADLIBs. Access to FOCADLIB EXEC is required.

## VM Command Line Support

The SERVER now will execute CP, CMS, and EXEC commands **while it is running**. To issue commands, simply enter the command on the command line as follows:

```
known_server_command
CP some_cp_command_and_options
EXEC some_exec_and_options
CMS some_cms_command_and_options
some_cms_command_and_options
```

---

If a command prefix (CP, CMS, or EXEC) is omitted, the default environment is CMS. To eliminate any ambiguity whether a command is a server or CMS command, always use a command prefix.

If a command is issued on the command line while the server is running, the command will be passed to CMS, just as if CMS *some\_command* had been typed in. This is done via native CMS support using CONIECB rather than immediate command support.

Commands are passed to the environment as an extended p-list, except that the commands and parameters are all uppercase. This allows ACCESS commands to SFS directories, FILEDEFS to data sets, and commands that require an extended p-list such as STDEBUG and STORMAP to be executed.

It should be noted that while the server is executing these commands, users may CONNECT and SEVER IUCV connections, but no other commands can be executed, until the SEVER returns from the operating system.

## Server Commands

Server commands are commands entered at the command line **while the server is running**. Enhanced server commands include:

Command	Function
? parameters	Query various settings on the server.
PARM parameters	Set/reset most configuration options.
CLEARSNK	Reset many control blocks back to NULL.
STOP [END]	Shutdown the server. The END option is needed if users are still active on the server and an immediate shutdown is required.
GO	Resets STOP command, if STOP has not yet completed.
HX	Shuts down the server like STOP END.
SHUTDOWN	Shuts down the server like STOP.

All of the above commands are intercepted by CMS immediate command support and are processed by an immediate command exit and an ECB POST.

## Server Query Commands

All query commands begin with a question mark (?). The following table lists valid query commands:

<b>Command</b>	<b>Function</b>
?	Displays list of valid query commands.
? CALLS [TOTALS ] [su_command]	Displays a list of all HLI calls with count. If TOTALS is specified, only the total number of HLI calls is displayed. If a specific HLI call is specified (such as OPN), then only the count for that command is displayed.
? CONNECT	Displays a list of users that have an IUCV CONNECT path defined.
? MONITOR	Displays current CP MONITOR activity statistics.
? PARM any_valid_config_command	Displays values of currently defined parm.
? QUEUES	Displays status of users currently queued.
? RELEASE	Displays release information of server.
? SCHEDULE	Displays list of users in the schedule list.
? SERVER	Displays status information on the server.
? SU	Same as the current ? SU command. For more information, see Listing Users on the Sink Machine: The ? SU Query.
? WHERE [module_name]	Displays load information on all modules, unless a specific module name is supplied.

---

## Server Parm Commands

Most configuration and command line options may be changed dynamically, while the server is running, with the OPTION command. Default of all options is “NO”. The syntax is:

*PARM most\_any\_valid\_config\_setting*

These parameters include:

ECHO NOECHO	ECHO records activity in a simple HLI PRINT file. NOECHO disables recording.
STAT NOSTAT	STAT records activity in an extended HLI PRINT file. NOSTAT disables recording.
VERBOSE NOVERBOSE	VERBOSE enables informational messages during the server operation. NOVERBOSE disables informational messages. VERBOSE and DEBUG are mutually exclusive.
DEBUG NODEBUG	DEBUG enables TRACE messages. These are issued each time a module is called. NODEBUG disables these messages. VERBOSE and DEBUG are mutually exclusive.
SMSG NOSMSG	SMSG enables the server to accept CP SMSG commands. NOSMSG disables SMSG support.
SNAP NOSNAP	SNAP enables SNAP DUMPS (SYMPTOM RECORDS) via DIAG x'94' at PROGRAM CHECK or SVC 13 (abend). NOSNAP disables this feature.
REIPL NOREIPL	REIPL issues a CP TRACE command that will IPL CMS at program check. This operation will occur before any program check handler. NOREIPL deletes the CP TRACE set.
MONITOR NOMONITOR MONCOUNT	MONITOR will cut CP MONITOR records, if server is authorized with OPTION APPLMON. If the server is not authorized, it will still monitor server activity, but this will not be sent to CP MONITOR. This can still be helpful for some server problems. MONNOCUT allows a user with APPLMON privileges to start monitor without sending these records to CP.
PASSWORD NOPASSWD	PASSWORD with no option is the same as defining no password. PASSWORD normally takes an option of a server-defined password or a special password. Special passwords include: ACIGROUP, LOGONPW, NOPASSWD. NOPASSWD may also be a separate command line option, which also resets the password to the default of having no password.

NOPASSWD	NOPASSWD resets the password back to the default of no server password.
GTFNSS NOGTFNSS	GTFNSS with no option is the same as defining no gtfNSS trace. GTFNSS normally takes an option of the name of the segment where TRACE records will be cut. NOGTFNSS disables cutting TRACE records to the segment.

## Configuration and Data Files

See configuration files for default values.

File	Description
IUCVMAIN CONFIG	A data file allows dynamic allocation of MAXUSERS and initial storage requirements. All of the command line options listed in <i>Starting a Sink Machine Using IUCV</i> on page 5-2 are supported in the CONFIG file.
CLNCNT	Defines the number of times the TIMER must POP with no IUCV activity before FCB cleanup occurs.
TIMER	The time in minutes for each TIMER POP. A setting of ZERO disables the timer and cleanup.
IBUFF	The initial IUCV buffer size in pages.
IFCBS	The initial number of FCBs defined at startup.
MFCBS	The maximum number of FCBs defined as active on the server at any time.
IUCVDCSS DATA	A data file created at gen time that contains the name, date, and time information of current segment definition.
FOCSUACC DATA	Defines a list of users and/or groups of users who are authorized to access the server via the ACIGROUP directory option. If omitted, all users are authorized. For more information, see <i>FOCSUACC DATA</i> on page 5-13.

---

# CP SMSG Support

The server can now accept CP SMSG for most commands. The following are examples of system commands:

```
CP SMSG server password CMD CMS LISTFILE * EXEC A
CP SMSG server password CMD CP IND USER
CP SMSG server * CMD EXEC DOTIME
CP SMSG server * CMD ? PARM
```

where:

*server*

Is the userid of the server.

*password*

Is the password specified when the server was started. A special password “LOGONPW” may have been specified when the server was started. If so, then the password must be the logon password of the server. DIAG x'A0' will be issued to validate the password.

Another special password, ACIGROUP, only allows users in the same ACIGROUP as the server to issue commands.

\*

Allows a list of “secure” users to issue commands without the need of entering a password. If \* is specified as the password, a GLOBALV list is searched for the user who issued the command. If the user is found in the list, the command is then executed.

Use the following syntax to set a list of users. This list may be up to 255 bytes long:

```
GLOBALV SELECT FOCIUCV SETL NOPASSWD id1 id2
```

CMD

Is a required keyword.

Any CMS, CP, EXEC, or server command can be executed. The server will use secondary console support to allow the user who issues the command to see the output on their terminal.

## NUCXLOAded in Storage

IUCVMMAIN, the new server module, has been generated with RLDDATA information. Because this module is self-nucxloading, it makes it easier to locate the module for debugging and should avoid possible conflicts with user modules that may be loaded at a fixed location, such as X'20000'.

## CP TRACE MC

To facilitate debugging, the MC (monitor call) instruction has been inserted at the beginning and end of most critical calls in the server communication modules.

To implement this trace, issue the following command:

```
CP TRACE MC RUN
```

This displays the trace to the terminal.

To suppress terminal output, issue the following command:

```
CP SPOOL PRINT TO *  
CP TRACE MC RUN NOTERM PRINT
```

This trace is extremely verbose. When enabled, it slows the server and increases CPU usage. Since MC is implemented as a hardware function, there is almost no overhead if not enabled. Usage should be limited and is *not* recommended for the general user.

## Segment TRACE

A trace of all calls to the server can be implemented using the GTFNSS option. This option causes 512 bytes of IUCV calling information to be written to an NSS segment. The size of the segment determines the number of entries that are written. A 1-megabyte segment holds 2,048 entries. A 20-megabyte segment holds 40,960 entries. This trace is very fast and has almost no overhead.

Use a command similar to the following to define a segment:

```
CP DEFSEG NSSTRACE 4800-4FFF EW
```

This defines the segment NSSTRACE as writable and 8 megabytes in length. Before usage, be sure to save the segment using the following command:

```
CP SAVESEG NSSTRACE
```

To use this segment, specify the following command line option:

```
GTFNSS NSSTRACE
```

Or, if the server is already running, issue the following:

```
PARM GTFNSS NSSTRACE
```

Definition of a trace segment normally requires CLASS E or a NAMESAVE option in the user's directory. This *cannot* be done by a general CLASS G user. Once defined, any user may use the segment.

---

## Automatic IPL

The command line option REIPL causes the server to issue the following CP TRACE commands:

```
CP TRACE PROG ID RIPL RUN CMD CP IPL CMS PARM AUTOOCR $ REIPL $
CP SET CONCEAL ON
```

This sets CONCEAL ON and defines a named trace command that will re-IPL the machine if a program check is encountered.

**Note:** The IPL command is limited to a generic CMS only.

The PARM \$ REIPL \$ is saved starting at location X'C0', by CMS, during the IPL. The PROFILE EXEC on the server should check this storage location and take any needed action, such as SEND A NOTE or SEND A MESSAGE.

The following REXX EXEC code fragment can be used to examine the IPL data. It *must* be examined in the PROFILE EXEC before other CMS programs overlay it.

```
parmdata = Translate(Storage('C0',64),' ','00'x)
```

A far better way to pass IPL PARMS to the PROFILE EXEC would be to make the following changes to the SYSPROF EXEC on the "S" disk:

```
Parse Arg insflags 'FF'x cp_keyword psw_spg hexloc 'FF'x ,
          sysid 'FF'x insparms 'FF'x
IF rc = 0 /*Did we find profile on disk accessed?*/
Then 'EXEC PROFILE' insparms
```

These changes must be made by the VM/CMS system programmers.

## FULL and SNAP Dumps

The command line option DUMP causes the following commands to be issued:

```
CP TRACE PROG ID DUMP RUN CMD CP VMDUMP 0:END DCSS FORMAT CMS *VMDUMP ON
SERVER DUE TO PROGRAM CHECK
```

The above is a single command and wraps because of length. This dump is forced to the server's READER.

The SNAP option causes a 'SYMPTOM' record to be cut, using DIAG x'94' during program check or abend processing by the program check and abend exit handlers.

## CP MONITOR

Support of analyzing data is built into the server code. The server will cut monitor records if the APPLMON option is included in the server VM directory entry. If APPLMON is omitted, records will be cut, on the server, but will not be transmitted to CP.

Velocity Software currently supports report writing using CANNED procedures for this data.

## User Exits

The following table lists user-written exits. All of the SVRXT(n) exits will attempt to dynamically load (OS LOAD SVC 8) the user exit once if undefined. This allows testing of exits without the need to regenerate the segment for each exit update. If the site does not include an exit, then each individual user may define an individual exit for their own server.

Exit	Description
SVRXT1	A user-written exit, defined as a WXTRN, that will be linked into the IUCVMAIN module at gen time if found on any accessed disk. If linked into IUCVMAIN, it will be called at initialization. If not linked into IUCVMAIN, an attempt will be made to load it dynamically at run time.
SVRXT2	A user-written exit, defined as a WXTRN, that will be linked into the IUCVMAIN module at gen time if found on any accessed disk. If linked into IUCVMAIN, it will be called at de-initialization. If not linked into IUCVMAIN, an attempt will be made to load it dynamically at run time.
SVRXT3	A user-written exit that is called at SVC 13, OS ABEND. This entry is defined as a WXTRN and will be loaded into the segment at HLI SEGMENT gen time if found on any accessed disk. All users of the HLI SEGMENT will access the same exit. An attempt will also be made to call this exit dynamically, but it is highly suggested that, if implemented, this exit be loaded into the segment at HLI SEGMENT gen time.
SVRXT4	A user-written exit that is called at PROGRAM INTERRUPT. This entry is defined as a WXTRN and will be loaded into the segment at HLI SEGMENT gen time if found on any accessed disk. All users of the HLI SEGMENT will access the same exit. An attempt will also be made to call this exit dynamically, but it is highly suggested that, if implemented, this exit be loaded into the segment at HLI SEGMENT gen time.

Exit	Description
SVRXT5	A user-written exit that is called similar to the CHARGE routine (see below). This entry will be loaded into the segment at HLI SEGMENT gen time if found on any accessed disk. All users of the HLI SEGMENT will access the same exit. If entry is not resolved, an attempt will be made to call this exit dynamically at run time.
SVRXT6	A user-written security exit installed in the segment that is called before and after any call to HLICTL. This entry is defined as a WXTRN and will be loaded into the segment at HLI SEGMENT gen time if found on any accessed disk. All users of the HLI SEGMENT will access the same exit. If entry is not resolved, an attempt will be made to call this exit dynamically at run time.
CHARGE	Currently loaded into the segment as a dummy routine and can be replaced by the user. At entry, the parmlist points to a full word of zero. At exit, the parmlist points to a full word of one.

## Security Considerations

This section describes the implications of using special passwords, the ability to issue commands without a password, and the FOCUSUACC DATA file.

### Special Passwords

If the server is started with the password *LOGONPW*, a call will be made to the security system via `DIAG x'A0'` to validate the password against the logon password of the server.

The server *must* be authorized to issue `DIAG x'A0'` or an abend will occur.

If the server is started with the password *ACIGROUP*, the *ACIGROUP* of the user and the *ACIGROUP* of the server *must* match before the user can issue a command. This allows all users of a specific *AGIGROUP* command authority for the server.

### Issuing Commands Without a Password

Those using “trusted” work stations can issue commands without the need for any password. If the password sent to the server is an asterisk (\*) a *GLOBALV* list is searched. If the user is found in the *GLOBALV* list, the command is issued. Define the *GLOBALV* list on the server using the following syntax:

```
GLOBALV SELECT FOCIUCV SETL NOPASSWD id1 id2 id3 ...
```

The list of userids can be up to 255 bytes long. A special user “ANYUSER” allows any user to issue SMSG commands to the server. This should be used for testing purposes only, not in a production environment.

## FOCSUACC DATA

The FOCSUACC DATA file contains specific users or specific ACIGROUPS that are authorized to access the server. Userids are limited only by virtual storage. ACIGROUP entries are limited to 16.

### Sample FOCSUACC DATA File

The following is a sample of the FOCSUACC DATA file that is shipped with FOCUS:

```
*
* This is a sample file. If FOCUSACC DATA is found on any accessed disk
* by the FOCUS SU SERVER, then ACCESS using security rules, defined in
* this file, will be used.
* This data file must be named FOCUSACC DATA, on any disk accessed by
* the SERVER.
*
* File MUST be RECFM=F LRECL=80.
*
* Comments begin with a '*' in column 1. Comments may follow entries
* so long as they begin in column 20. All userids, keywords, and groups
* must be in uppercase.
*
* Only users which are defined in this file or users in specific
* ACIGROUPs will be allowed access. Userids MUST begin in column 1.
*
* A special keyword "ACIGROUP" allows 'GROUPS' of users to be defined
* rather than specific userids. Entries can be 'mixed'. If ACIGROUP is
* defined, the group which accesses the server most should be defined
* BEFORE other ACIGROUPS. The 'GROUP' must follow the keyword, ACIGROUP
* and start in column 10. Entries are COLUMN sensitive.
*
VMUSERID   John Doe, dept sales, authorized by jack smith
VMUSERA    mary smith
VMUSERB    temp id for sales
VMUSERC
*
* allow all users with ACIGROUP of 'ACCT ' and 'MANAGE ' access
* where most users who access the server are in the 'ACCT ' group
*
ACIGROUP ACCT      * allow all accounting users to access
ACIGROUP MANAGE   * allow managers access
```

---

## EXEC Services

Several CMS EXECs will be executed on the server at specified intervals. All of the following EXECs are designed to be changed, modified, or deleted by the installation site. Using two EXECs allows a SITE and USER organization. None of the EXECs are necessary for SERVER operation.

<b>EXEC</b>	<b>Description</b>
IUSCLK00	Executes at midnight. This EXEC may shutdown the server if the exit return code is non-zero.
SRVCLK00	Executes after IUSCLK00 and only if IUSCLK00 returns a zero return code. SRVCLK00 may also shutdown the server if the return code is non-zero.
IUSCLK	Executes at the time interval specified in the IUCVMAIN CONFIG file (see TIMER setting). The server will also be shutdown if IUSCLK returns a non-zero return code.
SRVCLK	Executes after IUSCLK and only if IUSCLK returns a zero return code. The server will also be shutdown if SRVCLK returns a non-zero return code.
IUSCLN	Executed at cleanup only. This EXEC executes only after IUSCLK has executed 'nnn' times with no IUCV traffic. The 'nnn' setting is defined in IUCVMAIN CONFIG as CLNCNT.
SRVCLN	Executes after IUSCLN and only if IUSCLN returns a zero return code. The server will also be shutdown if SRVCLN returns a non-zero return code.

The following table lists EXECs that are used to generate the IUCV modules:

<b>EXEC</b>	<b>Description</b>
IUGSERV	Generates the IUCVMAIN module for IUCV.
IUGUSER	Generates user HLI modules.
IUGMAIN	Generates the HLIMAIN module for VMCF.

## SUBCOM for EXECs

Any EXEC executed under the server environment may issue server commands using the FOCIUCV or FOCFIFO subcommand environment. FOCIUCV will send output to the terminal and FOCFIFO will stack output FIFO. If output is stacked, the EXEC can then read the stack and process as needed.

Consider the following REXX code fragments:

```
instack = Queued()
'MAKEBUF'
buffnum = rc
Address 'FOCFIFO' '?' CALLS' /* display OPN CLO */
Do ix = 1 To Queued()-instack By 1
  Pull cmd cnt .
  If cmd = 'OPN' | cmd = 'CLO'
  Then Say "There were" cnt "calls to" cmd
End ix
'DROPBUF' buffnum
Address 'FOCIUCV' '?' SERVER' /* server stats */
```

## File Checkpoint

CMS FILE IO is normally buffered by the server. This allows the server to eliminate unneeded OPENS and CLOSEs for files that are already opened or closed. This feature is enabled by default on the server. To disable this feature, add the command line option NOMFD when the server is started or PARM NOMFD while the server is running.

**Note:** Disabling file checkpoint may slow the server.

## LOADLIB GLOBAL

At startup, the server will normally automatically GLOBAL any necessary LOADLIBs. This will replace any currently globalized loadlibs. If the current LOADLIB list is required to be available, the command line option NOGLOBAL may be specified. This forces the server not to change the current global list. It is up to the user to be sure the required FOCUS/HLI LOADLIBs are globalized or an abend will occur.

The easiest way to check the current list is to start up the server and issue the following syntax:

```
CMS QUERY LOADLIB
```

This will display the current required libraries.

You may also issue the GLOBAL LOADLIB command while the server is running. Be sure to include the required libraries in the list. Assuming MYLIB LOADLIB is to be globalized, the command would look like the following:

```
CMS GLOBAL LOADLIB FOCLIB FOCINT MYLIB
```

---

You may also use the IBI FOCADLIB EXEC to add user LOADLIBs. The following example adds USERACCT LOADLIB:

```
EXEC FOCADLIB LOADLIB USERACCT
```

This is the same as issuing

```
GLOBAL LOADLIB USERACCT
```

before starting the server and then starting the server with the GLOBUPDT option.

## Unsolicited Interrupts

The SERVER defines an unexpected external interrupt handler. If such an interrupt should occur, a message will be logged and processing will continue. The interrupt will be ignored.

Normally when CMS is presented with an unsolicited external interrupt, an abend would occur.

## FILEDEFS

The following table lists the FILEDEFS allowed:

<b>FILEDEF</b>	<b>Description</b>
HLIERROR	A device where severe error messages are written. If HLIERROR is not defined at startup, HLIERROR will automatically be defined.
HLIPRINT	A device where STAT or ECHO messages are written. Used by ECHO and STAT for output. If ECHO or STAT is specified, HLIPRINT will be automatically defined if not currently defined.
HLITERM	For internal use. If not defined at startup, HLITERM will be automatically defined.

## Sample Directory

The following is a simple user directory that has an ACIGROUP entry of FINANCE:

```
VMUSER1  DIRECT  A0 V 80  Trunc=72 Size=6 Line=1 Col=1 Alt=0
====>
00001 USER VMUSER1 LOGONPW 6M 31M G 64 ON ON ON ON
00002 ACCOUNT BOOKKEEP MAIL-B7
00003 *OWNER = Some VM User Name
00004 ACIGROUP FINANCE
00005 CONSOLE 009 3215 A
00006 MDISK 191 3390 1285 20 VOLSER1 M READPASS RITEPASS MULTPASS
00007 * * * End of File * * *
```

---

---

## APPENDIX A

# Improving Performance: Storing Central Databases On Disk

This appendix is addressed to systems support staff. It contains suggestions to improve the performance of Simultaneous Usage. Remember that the larger the application and the more users using it, the more important it is for the central database to be designed properly for maximum efficiency.

You can improve SU performance by storing central databases on disk in the following manner:

- Place different databases on different sink machines.
- Store central databases on disk packs that have little activity besides SU.
- Store the databases on volumes not used for paging or swapping, and do not store them on the SYSRES volume.
- Store each database on a separate volume. If this is not possible, distribute the databases across as many volumes as possible.
- Do not drive the volumes higher than their natural I/O rates. You can obtain the natural I/O rate for your device type from the appropriate IBM publication.
- Make sure that minidisks that are to contain central databases are formatted with a 4K blocksize.

---

## APPENDIX B

# FOCUS Error Messages

- (FOC500)      **SU. COMMAND NOT RECOGNIZED:**  
An invalid HLI command was issued. For example, NXT rather than the NEX command.
- (FOC501)      **SU. COMMAND SYNTAX INCORRECT:**  
Too few arguments have been provided in the HLI command.
- (FOC502)      **SU. FILE NOT OPENED:**  
The command cannot be executed because the file has not yet been opened (i.e., no OPN command has been issued).
- (FOC503)      **SU. MORE MEMORY NEEDED ON THE CENTRAL DATA BASE MACHINE:**  
The sink machine requires more virtual storage to operate. Restart the sink machine in a larger region.
- (FOC504)      **SU. MASTER FILE CANNOT BE LOCATED:**  
The requested Master File Description cannot be found by the sink machine.
- (FOC505)      **SU. DESCRIPTION OF DATA NOT FOUND:**  
The requested Master File Description cannot be found by the sink machine.
- (FOC506)      **SU. NO DATA FOUND FOR FILE:**  
No data found in the specified file. The file must first be initialized by the CREATE command even if no data is entered.
- (FOC507)      **SU. ERROR IN FILE DESCRIPTION:**  
An error was encountered in the Master File Description on the sink machine.
- (FOC508)      **SU. INVALID PARAMETER IN THE HLI CALL:**  
An invalid parameter was encountered in an HLI call received by the sink machine (for example, NTEST is less than zero).

- 
- (FOC509) **SU. FIELDNAME NOT FOUND IN FILE DESCRIPTION:**  
The fieldname referenced on a SHO command does not exist in the Master File Description. Check the spelling or the structure of the NAMES array passed into the SHO command.
- (FOC510) **SU. FILE IS NOT OPENED. CLOSE IGNORED:**  
The files specified in the CLO command is not open. The CLO command has been ignored.
- (FOC511) **SU. SEGMENT NAME NOT RECOGNIZED:**  
The segment specified as a TARGET or an ANCHOR segment is not found in the Master File Description.
- (FOC512) **SU. OPTION NOT SUPPORTED:**  
The HLI command passed to the sink is recognized but is not yet supported.
- (FOC513) **SU. NO CURRENT POSITION ACTIVE IN FILE:**  
No current position has been established from which to execute the command.
- (FOC514) **SU. TEST RELATION NOT RECOGNIZED:**  
An invalid test relation was passed to the sink machine. Possibly the number of tests was specified incorrectly. Valid test conditions are LT, LE, GT, GE, EQ, NE, CO, and OM.
- (FOC515) **SU. IMPROPER USE OF VIRTUAL SEGMENT:**  
An attempt was made to change or improperly use a cross-reference segment.
- (FOC516) **SU. ERROR ON INCLUDE. UNIQUE INSTANCE ALREADY EXISTS:**  
An attempt was made to include a second instance of a unique segment for a particular parent instance. The transaction is ignored.
- (FOC518) **SU. NO PATH EXISTS FROM THE 'ANCHOR' TO THE 'TARGET' SEGMENT":**  
The ANCHOR and TARGET segments specified do not lie on the same path in the file.
- (FOC519) **SU. INDEXED FIELD DOES NOT HAVE FIELDTYPE=I:**  
An error has occurred in the use of an indexed field, or a field named in an NXD command is not indexed.
- (FOC520) **SU. NO VALUE SUPPLIED FOR AN INDEXED LOOKUP:**  
On an NXD call, no tests on the target segment were provided.

- 
- (FOC521) **SU. NO CURRENT PARENT ESTABLISHED FOR CROSS-REFERENCE RETRIEVAL:**  
No parent position has been established for the retrieval of a cross-reference segment. The key for the linked segment is not active, and so no retrieval can be performed.
- (FOC522) **SU. THE PASSWORD DOES NOT PROVIDE FILE ACCESS RIGHTS:**  
Check the password provided in the FCB.
- (FOC523) **SU. PASSWORD DOES NOT ALLOW THIS ACTIVITY:**  
The command issued is not allowed with the current password. Check the password provided in the FCB.
- (FOC524) **SU. ATTEMPT TO INCLUDE A DUPLICATE SEGMENT:**  
The segment instance identified by the key values is already in the database. The database has not been changed. (Analogous to FOCURRENT=1 in MODIFY/SU.)
- (FOC525) **SU. CANNOT DELETE RANDOMLY OBTAINED SEGMENT:**  
Segment instances obtained with NXK or NXD may not be deleted.
- (FOC526) **SU. CURRENT POSITION HAS BEEN LOST:**  
Attempt to change the instance has not been performed, because the instance has been deleted by another user. (Analogous to FOCURRENT=2 in MODIFY/SU.)
- (FOC527) **SU. FCB IS STILL OPEN, AND CANNOT BE REUSED:**  
An OPN command was issued for an FCB which was already open. Before reusing the FCB, it must be closed.
- (FOC528) **SU. TOO MANY FILES OPEN ON THE CENTRAL DATA BASE MACHINE:**  
Too many files are open on the central database machine (limit 50) or too many FCBs are open (limit 256).
- (FOC529) **SU. CURRENT RECORD HAS BEEN CHANGED BY ANOTHER USER:**  
Attempt to change the instance has not been performed because it has already been changed by another user. (Analogous to FOCURRENT=3 in MODIFY/SU.)
- (FOC530) **SU. THE NUMBER OF FIELDS IN THE FILE EXCEEDS 1500:**  
SU cannot be used with more than 1,500 fields in a FOCUS file, including cross-references.
- (FOC531) **SU. ALL FILES IN COMBINE DO NOT HAVE SAME CENTRAL MACHINE:**  
The FOCUS USE command must designate that all of the files used in the COMBINE have the same sink machine. Either they must all be controlled by the same sink or they must all be local files.

- 
- (FOC535) **MORE CORE IS NEEDED TO EXECUTE REQUEST: COMMIT**  
The amount of main storage is insufficient for executing the specified command. Define larger storage and re-execute the command.
- (FOC538) **SU. CENTRAL AND LOCAL USERS HAVE DIFFERENT MASTER DESCRIPTIONS:**  
The master file description read by the central data base machine is not the same as the one used by the source or local machine.
- (FOC540) **SU. ENVIRONMENT PROBLEM IN THE CENTRAL DATA BASE MACHINE:**  
The central database machine has terminated because of a fatal error.
- (FOC541) **SU. CENTRAL DATABASE MACHINE ERROR:**  
A communication error has occurred on the central database machine.
- (FOC542) **SU. COMMUNICATION NOT AVAILABLE TO CENTRAL DATA BASE MACHINE:**  
The central database machine has not been started or the wrong name has been used for the central database machine.
- (FOC543) **SU. ERROR COMMUNICATING WITH CENTRAL DATA BASE MACHINE:**  
A communication error has occurred. Check to see that the sink machine is still active and, if not, restart it.
- (FOC544) **SU. COMMUNICATION NO LONGER AVAILABLE TO CENTRAL DATA BASE MACHINE:**  
The central data base machine is not active. Check to see if it has been terminated, and restart it.
- (FOC545) **SU. UNRECOVERABLE CENTRAL DATABASE MACHINE ERROR:**  
A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.
- (FOC546) **SU. UNRECOVERABLE CENTRAL DATABASE MACHINE ERROR:**  
A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.
- (FOC549) **INVALID. A SIMULTANEOUS USER HAS JUST INPUT THIS SEGMENT:**  
Another FOCUS user has input a segment with the same keys as you. Re-retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.
- (FOC550) **INVALID. A SIMULTANEOUS USER HAS JUST CHANGED THIS SEGMENT:**  
Another FOCUS user has changed some data in this segment before you. Re-retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.

- 
- (FOC551) **INVALID. A SIMULTANEOUS USER HAS JUST DELETED THIS SEGMENT:**  
Another FOCUS user has just deleted the segment you are trying to update or include. Re-retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.
- (FOC725) **ERROR IN HLI PROFILE:**  
Only certain SET commands may be issued in the HLI profile. Either a non-SET line was found, or an invalid SET command was issued. Check the file (PROFILE HLI on CMS or member HLIPROF of the FOCEXEC dataset on MVS). The erroneous line was ignored.
- (FOC726) **OPTION NOT SUPPORTED WITH COMMIT/SU:**  
COMMIT/SU does not support the use of an alternate file view.
- (FOC758) **NO FILE INTEGRITY AVAILABLE. MODIFY TERMINATED:**  
The COMMIT/ROLLBACK feature of MODIFY requires Absolute File Integrity in order to run.
- (FOC759) **FATAL ERROR DURING COMMIT. MODIFY TERMINATED:**  
A fatal error was encountered while applying the COMMIT to the database.
- (FOC760) **FATAL ERROR DURING ROLLBACK. MODIFY TERMINATED:**  
A fatal error was encountered while attempting to ROLLBACK the database.
- (FOC761) **UNABLE TO INITIALIZE FOCSORT. MODIFY TERMINATED**  
In order to run a MODIFY with the COMMIT/ROLLBACK feature, FOCSORT must be used. Check to see if FOCSORT was allocated properly.
- (FOC766) **UNABLE TO ESTABLISH SINK POSITION. ROLLBACK FORCED:**  
A position from which to match on the sink could not be established. Any further database actions are meaningless. Therefore a ROLLBACK is forced.
- (FOC767) **UNABLE TO ESTABLISH NEW COMMIT USER ON THE SINK:**  
There is no more room on the sink for another COMMIT user. As a result the user is terminated.
- (FOC769) **SU. FUNCTION DISABLED: COMMIT:**  
COMMIT functionality on the sink machine has been disabled. Contact your sink machine administrator.
- (FOC896) **READ OPTION REQUIRED ON USE STATEMENT WITH AS AND ON OPTIONS**  
Must specify the READ option on a USE statement when using the AS and ON options.

---

---

## APPENDIX C

# HLI Status Codes Returned in FCB Word 24

<b>Return Code</b>	<b>Meaning</b>
0	Command executed normally.
1	Command executed normally, but no segments were retrieved. The current position is unchanged. Either the qualifying conditions failed to locate the desired record, or an end-of-chain condition occurred. (i.e., no more target segment instances within the anchor segment).
760	Command not recognized. An invalid HLI command was issued. For example, NXT rather than the NEX command.
761	Too few arguments have been provided in the HLI command.
762	The command cannot be executed because the file has not yet been opened (i.e., no OPN command has been issued).
763	The sink machine requires more virtual storage to operate. Restart the sink machine in a larger region.
764	The requested Master File Description cannot be found by the sink machine.
765	The requested Master File Description cannot be found by the sink machine.
766	No data found in the specified file. The file must first be initialized by the CREATE command even if no data is entered.
767	An error was encountered in the Master File Description on the sink machine.
768	An invalid parameter was encountered in an HLI call received by the sink machine (for example, NTEST is less than zero).
769	The fieldname referenced on a SHO command does not exist in the Master File Description. Check the spelling or the structure of the NAMES array passed into the SHO command.

<b>Return Code</b>	<b>Meaning</b>
770	The file specified in the CLO command is not open. The CLO command has been ignored.
771	Segment name not recognized. The segment specified as a TARGET or an ANCHOR segment is not found in the Master File Description.
772	The HLI command passed to the sink is recognized but is not yet supported.
773	No current position has been established from which to execute the command.
774	Test relation not recognized. An invalid test relation was passed to the sink machine. Possibly the number of tests was specified incorrectly. Valid test conditions are LT, LE, GT, GE, EQ, NE, CO, and OM.
775	Improper use of virtual segment. An attempt was made to change or improperly use a cross-reference segment.
776	An attempt was made to include a second instance of a unique segment for a particular parent instance. The transaction is ignored.
778	The ANCHOR and TARGET segments specified do not lie on the same path in the file.
779	An error has occurred in the use of an indexed field, or a field named in an NXD command is not indexed.
780	On an NXD call, no tests on the target segment were provided.
781	No parent position has been established for the retrieval of a cross-reference segment. The key for the linked segment is not active, and so no retrieval can be performed.
782	The password does not provide file access rights. Check the password provided in the FCB.
783	The command issued is not allowed with the current password. Check the password provided in the FCB.
784	The segment instance identified by the key values is already in the database. The database has not been changed. (Analogous to FOCURRENT=1 in MODIFY/SU.)
785	Segment instances obtained with NXX or NXD may not be deleted.

<b>Return Code</b>	<b>Meaning</b>
786	Attempt to change the instance has not been performed, because the instance has been deleted by another user. (Analogous to FOCURRENT=2 in MODIFY/SU.)
787	An OPN command was issued for an FCB which was already open. Before reusing the FCB, it must be closed.
788	Too many files are open on the central database machine (limit 50) or too many FCBs are open (limit 256).
789	Attempt to change the instance has not been performed because it has already been changed by another user. (Analogous to FOCURRENT=3 in MODIFY/SU.)
790	SU cannot be used with more than 1,500 fields in a FOCUS file, including cross-references.
791	The FOCUS USE command must designate that all of the files used in the COMBINE have the same sink machine. Either they must all be controlled by the same sink or they must all be local files.
798	The master file description read by the central data base machine is not the same as the one used by the source or local machine.
800	The central database machine has terminated because of a fatal error.
801	A communication error has occurred on the central database machine.
802	The central database machine has not been started or the wrong name has been used for the central database machine.
803	A communication error has occurred. Check to see that the sink machine is still active and, if not, restart it.
804	The central data base is not active. Check to see if it has been terminated, and restart it.
805	A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.
806	A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.

---

# Index

## #

#CP DISCONN command, 2-6

## ?

? SU query command

IUCV and, 5-5

RETCODE values, I-6

## A

Absolute file integrity, I-13

Automatic IPL, 5-10

## B

Buffer

Writing transactions from, I-5

## C

Central database storage, YY-1

Centrally controlled databases

Closing, I-5

Local databases and, I-4

Change verify protocol, 1-4

Checkpoint facility, I-12

COMMIT and, I-18

ROLLBACK and, I-18

CMS FILEDEF, 2-12

CMS PRINT command, 2-14

COMMIT, I-13

Checkpoint facility and, I-18

Seeing user changes, I-19

Sink Machine and, I-14

Subcommands, I-13

CP DISCONN, 2-2

CP SET RUN ON, 2-2

CP TRACE MC, 5-9

CRTFORM

Validating fields, I-9

## E

ECHO parameter, 2-2

Error codes, I-4

Error messages, I-3, B-1

Extended HLIPRINT trace, 2-14

## F

FCB, I-2

Centrally controlled databases

Local databases and, I-4

Error codes, I-4

Layout, I-3

Local databases

Centrally controlled databases and, I-4

File Communication Block. See FCB

FILEDEF SYSIN TERM, 2-2

FOCURRENT variable, I-7

Using, I-18

FOCUS GENHLI command, I-2

FOCUSSU

Commands

CP DISCONN, 2-2

CP SET RUN ON, 2-2

FILEDEF SYSIN TERM, 2-2

HLIMAIN, 2-2

## H

HLIERROR file, 2-3, 2-12

HLIMAIN, 2-2

HLIPRINT file, 2-12  
 CMS PRINT command, 2-14  
 Contents, 2-8  
   Columns, 2-8  
 Contents (extended format), 2-10  
   Additional columns, 2-10  
 ECHO option, 2-14  
 Extended format, 2-12  
 Extended HLIPRINT trace and, 2-14  
 Logging activity on, 2-7  
 NEW command and, 2-10  
 STAT option, 2-14  
 Uses of, 2-7

Host Language Interface (HLI), 1-3, I-1  
 Centrally controlled databases  
   Closing, I-5  
   Local databases and, I-4  
 Control commands, I-5  
 Creating an HLI module, I-2  
 FCB, I-2  
   Error codes, I-4  
 Return codes, I-1  
 Stopping a sink machine, I-5  
 VMCF, I-6  
 Writing transactions from buffer, I-5

## I

Inter User Communication Vehicle. See IUCV

IUCV, 1-2, 5-1  
 Configuration and data files  
   CLNCNT, 5-7  
   FOCUSUACC DATA, 5-7  
   IBUFF, 5-7  
   IFCBS, 5-7  
   IUCVDCSS DATA, 5-7  
   IUCVMAIN CONFIG, 5-7  
   MFCBS, 5-7  
   TIMER, 5-7  
 CP SMSG support, 5-8  
   Automatic IPL, 5-10  
   CP MONITOR, 5-11  
   CP TRACE MC, 5-9  
   FULL and SNAP dumps, 5-10  
   NUCXLOAded in storage, 5-8

Segment TRACE, 5-9  
 User exits, 5-11  
 EXEC services, 5-14  
   SUBCOM, 5-15  
 File checkpoint, 5-15  
 FILEDEFS  
   HLIERROR, 5-16  
   HLIPRINT, 5-16  
   HLITERM, 5-16  
 LOADLIB GLOBAL, 5-15  
 Overview, 5-1  
 Sample Directory, 5-17  
 Security considerations  
   FOCUSUACC DATA, 5-13  
   Issuing commands without a password, 5-12  
   Special passwords, 5-12  
 Server command line options  
   DEBUG/NODEBUG, 5-2  
   DUMP, 5-10  
   FULL, 5-10  
   GLOBAL/NOGLOBAL, 5-3  
   GLOBUPDT, 5-3  
   GTFNSS/NOGTFNSS, 5-3  
   MFD/NOMFD, 5-3  
   MONITOR, 5-3  
   MONNOCUT, 5-3  
   PASSWORD/NOPASSWD, 5-3  
   REIPL/NOREIPL, 5-2  
   SMSG/NOSMSG, 5-2  
   SNAP, 5-10  
   SNAP/NOSNAP, 5-2  
   VERBOSE/NOVERBOSE, 5-2  
 Server commands  
   CLEARSNK, 5-4  
   GO, 5-4  
   HX, 5-4  
   OPTION, 5-4  
   Parm, 5-6  
   Query commands, 5-5  
   SHUTDOWN, 5-4  
   STOP, 5-4

## Server query commands

- ? , 5-5
- ? CALLS, 5-5
- ? CONNECT, 5-5
- ? MONITOR, 5-5
- ? OPTIONS, 5-5
- ? QUEUES, 5-5
- ? RELEASE, 5-5
- ? SCHEDULE, 5-5
- ? SERVER, 5-5
- ? SU, 5-5
- ? WHERE, 5-5

Unsolicited interrupts, 5-16

VM command line support, 5-3

**L**

## Local databases

Centrally controlled databases and, I-4

LOCAL keyword, I-3

Multi-Threaded Reporting Facility and, 2-17

**M**

MAINTAIN, 1-3, 2-7, I-6

Protecting sink transactions, I-12

Checkpoint facility, I-12

COMMIT, I-13

ROLLBACK, I-13

Sample request, 1-5

MATCH command, I-9

MODIFY, 1-3, 2-7, I-6

Protecting sink transactions, I-12

Checkpoint facility, I-12

COMMIT, I-13

ROLLBACK, I-13

Sample request, 1-4

Multi-Threaded Reporting Facility, 2-16

FCB setup, 2-18

HLI load module, 2-18

Linking, 2-18

LOCAL keyword, 2-17

Shadow writing, 2-17

Special considerations, 2-20

USE command, 2-17

**N**

NEW command, 2-10

**O**

ON

AS and, I-2

**P**

PROFILE HLI. See SU profile

**R**

RETCODE values, I-6

ROLLBACK, I-13

Checkpoint facility and, I-18

Seeing user changes, I-19

Sink Machine and, I-14

Subcommands, I-13

**S**

Segment TRACE, 5-9

Server parm commands

Parameters

DEBUG/NODEBUG, 5-6

Server Parm commands

Parameters

ECHO/NOECHO, 5-6

GTFNSS/NOGTFNSS, 5-7

MONITOR/NOMONITOR/MONCOUNT, 5-6

PASSWORD/NOPASSWD, 5-6

REIPL/NOREIPL, 5-6

SMSG/NOSMSG, 5-6

SNAP/NOSNAP, 5-6

STAT/NOSTAT, 5-6

VERBOSE/NOVERBOSE, 5-6

SESSCM ASSEMBLE, 2-4

SET PATHCHECK command, I-20

Shadow writing, 2-17

Simultaneous Usage

Environment, 1-2

FOCUS language and, I-1

Improving performance, YY-1

IUCV and, 5-1

Overview, 1-2

Sink Machine, 1-2, 2-1

Sink Machine, 1-2, 2-1

? SU query, I-4

Testing status of, I-6

COMMIT and, I-14

Controlling processing, I-19

Controlling currency checking

SET PATHCHECK command, I-20

Error messages, I-3

Increasing maximum number of users, 2-4

Listing users, I-4

Logging activity on the HLIPRINT file, 2-7

Protecting transactions, I-12

ROLLBACK and, I-14

SESSCM ASSEMBLE, 2-4

Starting, 2-2

Stopping, 2-4, I-5

? Reply, 2-5

Continuing SU mode, 2-6

HX reply, 2-6

STOP reply, 2-5

SU profile, 2-3

STAT parameter, 2-2

SU profile

Commands

SET BINS, 2-3

SET CACHE, 2-3

COMMIT, I-19

ROLLBACK, I-19

## T

TABLE, 2-7

Transactions

Writing from buffer, I-5

## U

USE command, I-2

Multi-Threaded Reporting Facility and, 2-17

ON with AS, I-2

## V

Virtual Machine Communication Facility. See VMCF

VMCF, 1-2, 2-16, I-6

# Reader Comments

---

---

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. Send comments to:

Corporate Publications  
Attn: Manager of Documentation Services  
Information Builders  
Two Penn Plaza  
New York, NY 10121-2898

or FAX this page to (212) 967-0460, or call **Sara Elam** at (212) 736-4433, ext. **3207**.

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

Telephone: \_\_\_\_\_ Date: \_\_\_\_\_

Comments:

# Reader Comments

---