

# **FOCUS for S/390**

Simultaneous Usage for OS/390 & MVS  
User's Manual

# Contents

<b>1</b>	<b>Introduction .....</b>	<b>1-1</b>
	What is Simultaneous Usage? .....	1-2
	How SU Processes Transactions Without COMMIT and ROLLBACK .....	1-3
	The Multi-Threaded SU Reporting Facility .....	1-5
<b>2</b>	<b>Operating the FOCUS Database Server .....</b>	<b>2-1</b>
	Starting a FOCUS Database Server .....	2-2
	Using the SU Profile .....	2-5
	Stopping the FOCUS Database Server .....	2-6
	Logging FOCUS Database Server Activity in HLIPRINT .....	2-7
	HLIPRINT File Contents .....	2-8
	HLIPRINT File Contents (Extended Format) .....	2-10
	Producing HLIPRINT Reports .....	2-12
	Calculating Core Requirements for the FOCUS Database Server .....	2-14
	Protecting FOCUS Databases: the FOCUS/SU Security Interface (SUSI) .....	2-16
	Using the SU Security Interface .....	2-16
	How the FOCUS Database Server Invokes the Security Interface .....	2-16
	How the FOCUS Database Server Checks Access Rules .....	2-17
	How the FOCUS Database Server Manages the Number of Users .....	2-17
	Error Messages .....	2-19
<b>3</b>	<b>SU and The FOCUS Language .....</b>	<b>3-1</b>
	Using Centrally Controlled Data Sources .....	3-2
	Using the Multi-threaded SU Reporting Facility .....	3-3
	Special Considerations for Using Multi-threaded SU Reporting .....	3-5
	Messages From the FOCUS Database Server .....	3-5
	Error Messages .....	3-5
	Listing FOCUS Database Server Users .....	3-6
	Status Testing with ? SU Queries .....	3-7
	Using MODIFY or Maintain With SU .....	3-7
	Evaluating Conflicts: The FOCURRENT Variable .....	3-8
	Testing for Rejected Transactions .....	3-9
	Validating CRTFORM Turnaround Fields .....	3-9
	Protecting FOCUS Database Server Transactions .....	3-12
	The Checkpoint Facility .....	3-12
	Managing Transactions: COMMIT and ROLLBACK .....	3-13
	COMMIT and ROLLBACK Subcommands .....	3-13
	Using COMMIT and ROLLBACK With the FOCUS Database Server .....	3-14

SU Performance Considerations.....	3-17
Comparing COMMIT and ROLLBACK to the Checkpoint (CHECK) Facility.....	3-18
Using FOCURRENT .....	3-18
Viewing Other Users' Changes While Using COMMIT and ROLLBACK .....	3-18
Using Commands in the SU Profile for COMMIT and ROLLOACK .....	3-19
Controlling COMMIT Processing on the FOCUS Database Server. ....	3-19
Controlling Currency Checking on the FOCUS Database Server .....	3-19
<b>4 Host Language Interface (HLI).....</b>	<b>4-1</b>
Gaining Access to FOCUS Database Servers.....	4-2
The File Communication Block (FCB) .....	4-2
FCB Area Layout .....	4-3
Using Both Local and Centrally Controlled Databases .....	4-3
HLI Error Codes .....	4-4
Closing Centrally Controlled Databases.....	4-4
Writing Transactions from the Buffer .....	4-5
Using HLI Control commands .....	4-5
<b>A Improving SU Performance - Technical Notes .....</b>	<b>A-1</b>
Storing Central Databases on Disk .....	A-2
Additional Suggestions.....	A-2
<b>B Error Messages .....</b>	<b>B-1</b>
Accessing Errors Files .....	B-2
Displaying Messages Online .....	B-2
<b>C HLI Status Codes Returned in FCB Word 24 .....</b>	<b>C-1</b>
HLI Status Codes.....	C-2
<b>Index .....</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, WebFOCUS, and WorldMART 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/390, 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, SunOS, and iPlanet 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.

Allaire and JRun are trademarks of Allaire Corporation.

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

This documentation describes how to use the Simultaneous Usage facility in the OS/390 and MVS environments. It is intended for FOCUS users and administrators of centrally controlled data sources. It is available in the following formats: printed manual, PDF, and CD-ROM.

## How This Manual Is Organized

This manual includes the following chapters:

<b>Chapter/Appendix</b>	<b>Contents</b>	
<b>1</b>	<i>Introduction</i>	Introduces Simultaneous Usage terms and concepts.
<b>2</b>	<i>Operating the FOCUS Database Server</i>	Describes the operation of the FOCUS Database Server, which controls centrally accessed data sources.
<b>3</b>	<i>SU and the FOCUS Language</i>	Describes how to issue FOCUS requests against centrally controlled data sources.
<b>4</b>	<i>Host Language Interface (HLI)</i>	Discusses how to operate SU using HLI.
<b>A</b>	<i>Improving SU Performance - Technical Notes</i>	Describes techniques and suggestions for improving SU performance.
<b>B</b>	<i>Error Messages</i>	Describes how to access ERRORS files and to generate error messages online.
<b>C</b>	<i>HLI Status Codes Returned in FCB Word 24</i>	Lists HLI status codes.

## Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must enter exactly as shown.
<code>this typeface</code>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, indicates a cross-reference, or emphasizes an important term.
<b>this typeface</b>	Highlights file names and commands (in a text paragraph) that must be lowercase.
{ }	Indicates two choices from which you must choose one. You type one of these choices, not the braces.
[ ]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the information within the brackets, not the brackets.
	Separates two mutually exclusive choices in a syntax line. You type one of these choices, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameters, not the ellipsis points (...).
. . . . . .	Indicates that there are (or could be) intervening or additional commands.

## Related Publications

See the *Information Builders Technical Publications Catalog* for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969-4636.

You can also visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order.

# Customer Support

Do you have questions about the Simultaneous Usage facility?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your CMS Installation Guide questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of [www.informationbuilders.com](http://www.informationbuilders.com) also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

## Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (xxxx.xx).
- The FOCEXEC procedure (preferably with line numbers).
- Master File with picture (provided by CHECK FILE).
- Run sheet (beginning at login, including call to FOCUS), containing the following information:
  - ? RELEASE
  - ? FDT
  - ? LET
  - ? LOAD
  - ? COMBINE
  - ? JOIN
  - ? DEFINE
  - ? STAT
  - ? SET/? SET GRAPH
  - ? USE

- ? TSO DDNAME
- The exact nature of the problem:
  - Are the results or the format incorrect; are the text or calculations missing or misplaced?
  - The error message and code, if applicable.
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of CMS are you using? Has it, FOCUS, your security system, or an interface system changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

## 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. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Thank you, in advance, for your comments.

## 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 information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

---

## CHAPTER 1

# Introduction

### Topics:

- What is Simultaneous Usage?
- How SU Processes Transactions Without COMMIT and ROLLBACK
- The Multi-Threaded SU Reporting Facility

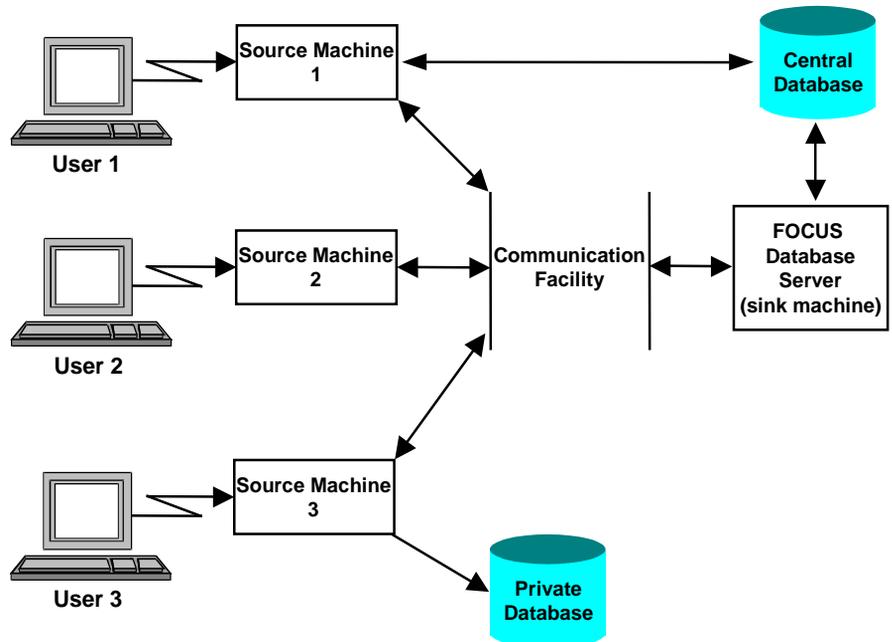
This chapter introduces Simultaneous Usage, a facility that enables multiple users to update a FOCUS database at the same time. The first section defines Simultaneous Usage and the second explains how it works.

# What is Simultaneous Usage?

Multiple users can read and change a FOCUS database at the same time, using FOCUS and/or Host Language Interface (HLI) commands, through the Simultaneous Usage (SU) facility. Without SU, only one user can update a database at a time - even databases that are allocated for sharing (DISP=SHR).

With SU, a centrally controlled database is allocated to a background job called the FOCUS Database Server. Database service requestors' TSO IDs, MSO sessions, and batch jobs running FOCUS, as well as programs using HLI are all called source machines. User's source machines send requests and transactions to the FOCUS Database Server, which processes their transactions and transmits retrieved data and messages back to the source machines.

The following representation of SU shows three source machines, a TSO user, a batch job, and an HLI application, all executing FOCUS requests. Source machines communicate with the FOCUS Database Server through cross-memory posting. Messages traveling between a source machine and the FOCUS Database Server are placed in the MVS Common Storage Area (CSA), which is accessible to both machines. When the FOCUS Database Server receives a request from a source machine, it changes or retrieves data from the centrally controlled database and transmits the results back to the source machine. Notice that User 3 on the diagram is also working on a locally controlled database (not managed by SU).



*Components of the Simultaneous Usage Environment*

SU is not needed when users are simply reading a database (for example, entering TABLE requests), but is needed when they want to change databases that others are reading.

Only those changing databases need SU; the Multi-Threaded SU Reporting Facility permits multiple users to read centrally controlled databases without a FOCUS Database Server. The Multi-Threaded SU Reporting Facility is described later in this chapter.

Note the following:

- SU enables up to 256 Source machines (users and batch jobs) to change a data sources on a single server concurrently. You can increase this number during FOCUS/SU installation.
- The FOCUS Database Server and source machines must all be running the same version and release level of FOCUS.
- A single source machine can simultaneously communicate with up to eight FOCUS Database Servers.

## How SU Processes Transactions Without COMMIT and ROLLBACK

When you submit a transaction that changes a segment, whether with a MODIFY, a Maintain, or an HLI program, SU employs a procedure called change/verify protocol. If more than one user tries to change a segment instance at the same time, change/verify protocol determines which transaction is accepted. A description of the change/verify protocol follows:

1. Identify the database instance you wish to change. In FOCUS, do this with MATCH or NEXT statements in a MODIFY or Maintain request. In HLI, use HLI commands in your program. Your source machine forwards the identifying values to the FOCUS Database Server, which uses the values to retrieve the correct instance.
2. The FOCUS Database Server retrieves the original database instance and saves it, while sending a copy back to the source machine.
3. Your MODIFY request or HLI program indicates changes 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.
4. Your source machine sends the updated copy back to the FOCUS Database Server.

5. The FOCUS Database Server then checks to see that no other user changed the instance in the database by comparing the current value in the database to the value saved in Step 2.

If the current value matches the original value, no one has changed the instance, and FOCUS changes the instance in the database using the copy from your source machine.

If the values differ, the FOCUS Database Server signals a conflict and rejects the source machine copy, because another user who requested the same instance at the same time has already changed it.

The following example illustrates change-verify protocol by tracing the execution of a FOCUS MODIFY request. The same concepts apply to HLI programs.

```
MODIFY FILE EMPLOYEE
PROMPT EMP-W PAY-DATE
MATCH EMP-W
  ON NOMATCH REJECT
  ON MATCH CONTINUE
MATCH PAY-DATE
  ON NOMATCH REJECT
  ON MATCH TYPE "CURRENT GROSS:<D.GROSS"
  ON MATCH PROMPT GROSS
  ON MATCH UPDATE GROSS
  ON MATCH TYPE "UPDATED!"
DATA
```

When you execute this, you are prompted for an employee ID and pay date. Enter:

```
EMP-ID   = 071382660
PAY-DATE = 990831
```

The source machine forwards the employee ID value to the FOCUS Database Server, which retrieves the instance with this ID and saves it. It also makes a copy for the source machine. When the source machine receives its copy, it forwards the pay date to the FOCUS Database Server, which retrieves the segment instance with PAY-DATE = 990831 and GROSS = 916.67.

The FOCUS Database Server saves these values and sends a copy to your source machine, which then prompts you for a new GROSS value, to which you respond:

```
GROSS = > 1050.35
```

Your machine updates its copy of the instance with the new GROSS value, 1050.35.

SU sends your new value to the FOCUS Database Server, which compares the current database value with the original value you retrieved (916.67). If another user already changed the value and they do not match, your transaction would be rejected and you would get an error message.

When several users try to change the same instance in a centrally controlled database at the same time, FOCUS accepts only the first transaction, rejecting all others on the assumption that if aware of the other user's changes they might not wish to make it. Users can always repeat the transaction later if they do. By reentering the same key values, they will receive a fresh copy of the instance and can update that.

Change/Verify protocol retains only the current transaction. When you rematch on the keys the current transaction becomes the new match subject. For example, when you retrieve data in one MODIFY case and then update it in another; verification is only performed on the second MATCH operation. Similarly, only the last record retrieved in multi-record processing (through a HOLD in MODIFY or a FOR NEXT in MAINTAIN) is retained. The current position in the database changes with each record retrieval or update

## **The Multi-Threaded SU Reporting Facility**

FOCUS can read and report from centrally controlled databases directly through the operating system using the Multi-Threaded SU Reporting facility. By using direct access instead of going through a FOCUS Database Server, you greatly reduce the access time required to process TABLE requests and produce reports.

---

## CHAPTER 2

# Operating the FOCUS Database Server

### Topics:

- Starting a FOCUS Database Server
- Stopping the FOCUS Database Server
- Logging FOCUS Database Server Activity in HLIPRINT
- Calculating Core Requirements for the FOCUS Database Server
- Protecting FOCUS Databases: the FOCUS/SU Security Interface (SUSI)
- Error Messages

This chapter describes how to operate the FOCUS Database Server (the job that handles updates to centrally-controlled databases). It is addressed to system and database administrators and others and discusses

After starting a FOCUS Database Server job, it does not require additional attention until you wish to stop it. Once it is running, simultaneous users can safely read and modify the databases under its control.

## Starting a FOCUS Database Server

Before beginning simultaneous usage (SU) of centrally controlled FOCUS data sources, the system administrator must start the FOCUS Database Server. Once started, this job remains in a wait state until FOCUS applications give it work to perform.:

The FOCUS Database Server executes a program called HLISNK after allocating the following files:

- The FOCLIB.LOAD data set is allocated to both ddnames STEPLIB and FOCLIB.
- The ERRORS.DATA dataset is allocated to ddname ERRORS.
- The communication data set is allocated to ddname FOCUSU. This data set is allocated to the FOCUS Database Server with a disposition of share (DISP=SHR). This communication data set introduces the FOCUS Database Server and source machines to each other through a handshake, enabling communication between them. Following the initial handshake all communications between the FOCUS Database Server and source machines take place in virtual storage.

**Note:** You create the communication data set when you install SU at your site. Each FOCUS Database Server requires a unique communication data set, so users planning to run more than one, will require a different communication data set for each. Give these data sets the following attributes:

`LRECL=16, BLKSIZE=16, RECFM=F, SPACE=1 TRK`

- Allocate a dummy FOCUS database to ddname FOCUSSU, with the following attributes:

`LRECL=4096, BLKSIZE=4096, RECFM=FB, SPACE=1 track`

Allocate it with the CREATE command at installation time to the FOCUS Database Server (not to the source machines) giving it a disposition of SHR. The database can be allocated to any number of FOCUS Database Servers and it is recommended that you recreate it with each new release of FOCUS.

To format the file, allocate it to ddname FOCUSSU, allocate MASTER.DATA to ddname MASTER, start a FOCUS session and issue the CREATE command:

## Syntax

### Allocating the FOCUSSU Dummy Database

CREATE FILE FOCUSSU

- The FOCUS Database Server writes diagnostic information to the ddname HLIERROR in the event of Database Server problems. HLIERROR should always be allocated to a SYSOUT class rather than to a data set.
- When using the HLIPRINT facility allocate a sequential data set or SYSOUT to ddname HLIPRINT to record user transactions.
- When using the COMBINE command, allocate a work file to ddname FOCSORT.
- The data sources that the FOCUS Database Server will process. If you wish to use the Multi-Threaded SU Reporting Facility, you must allocate these data sources with DISP=SHR.
- Allocate the partitioned data sets containing the database Master Files, concatenated to ddname MASTER. One of these PDSs must contain the member FOCUSSU. This Master File Description can be found in the data set MASTER.DATA.  
**Note:** Only data sets with identical LRECL and RECFM formats should be concatenated. While there is no restriction on concatenating data sets with different block sizes, the data set with the largest block size must always be named first in the concatenation.
- Allocate the partitioned data set containing the HLI profile to ddname FOCEXEC.

## Example

## Running the FOCUS Database Server

Here is sample JCL for running the FOCUS Database Server:

```
//*      Job card goes here
//HLISNK  EXEC PGM=HLISNK,PARM='parameters - see below'
//STEPLIB DD  DSN=FOCLIB.LOAD,DISP=SHR
//FOCLIB  DD  DSN=FOCLIB.LOAD,DISP=SHR
//ERRORS  DD  DSN=ERRORS.DATA,DISP=SHR
//*
//*      FOCUSU IS THE COMMUNICATION FILE
//*
//FOCSU   DD  DSN=FOCSU.DATA,DISP=SHR
//*
//*      FOCUSU IS A DUMMY FOCUS DATABASE CREATED DURING SU INSTALLATION
//*
//FOCUSSU DD  DSN=FOCUSSU.FOCUS,DISP=SHR
//*
//*      FOCSORT IS NEEDED IF THE COMBINE/SU FEATURE WILL BE USED
//*
//FOCSORT DD  SPACE=(TRK,(1,1)),UNIT=SYSDA
//*
//*      HLIERROR IS USED FOR DIAGNOSTICS
//*
//HLIERROR DD  SYSOUT=*,DCB=(LRECL=80,RECFM=F,BLKSIZE=80)
//*
//*      HLI PRINT IS THE LOG FILE (HERE, THE SYSTEM PRINTER)
//*
//HLI PRINT DD  SYSOUT=*.DCB=(LRECL=88,RECFM=FB,BLKSIZE=8800)
//*      or DD  SYSOUT=*.DCB=(LRECL=133,RECFM=FBA,BLKSIZE=13300)
//*
//*      MASTER CONTAINS THE FILE DESCRIPTIONS OF ALL
//*      DATABASES USED IN SU MODE. THE MASTER FOR FOCUSU
//*      IS IN MASTER.DATA. ONLY DATA SETS WITH IDENTICAL LRECL
//*      AND RECFM ATTRIBUTES SHOULD BE CONCATENATED
//*
//MASTER  DD  DSN=DATAID.MASTER.DATA,DISP=SHR
//*      DD  DSN=MASTER.DATA,DISP=SHR
//*
//*      FOCEXEC IS FOR THE SU PROFILE
//*
//FOCEXEC DD  DISP=SHR,DSN=DATDD.FOCEXEC.DATA
//*
//*      HERE ARE THE ACTUAL CENTRALLY CONTROLLED FOCUS
//*      DATABASES ALLOCATED WITH DISP=SHR SO MULTI-THREADED
//*      SU REPORTING FACILITY CAN BE USED.
//*
//EMPLOYEE DD  DSN=EMPLOYEE.FOCUS,DISP=SHR
//EDUCFILE DD  DSN=EDUCFILE.FOCUS,DISP=SHR
//JOBFILE  DD  DSN=JOBFILE.FOCUS,DISP=SHR
```

The PARM field in the HLISNK EXEC statement is optional; include it if you plan to use the SU ECHO facility, or if you plan to use HLI control commands).

## Syntax

### How to Specify the PARM Field on the HLISNK EXEC Card

```
PARM= '[password] {ECHO|STAT}'
```

where:

```
ECHO | STAT
```

Activate the SU HLIPRINT facility. Both parameters are optional. If the password is omitted, a leading blank must precede the word ECHO or STAT, and you cannot use HLI control commands. ECHO produces a standard format HLIPRINT report. STAT produces an extended format report. See *Logging FOCUS Database Server Activity in HLIPRINT* on page 2-7 for discussions of these reports.

## Using the SU Profile

The Simultaneous Usage Profile (SU Profile) enables users to set parameters for the FOCUS Database Server in a profile for use in controlling a local HLI program. This profile is member HLIPROF in a PDS allocated to the ddname FOCEXEC in the FOCUS Database Server. The DCBs are the same as those for any FOCEXEC PDS.

You can include the following commands in the FOCUS Database Server profile:

```
SET BINS = nn
```

```
SET CACHE = nn
```

```
SET SUTABSIZE = nnn
```

```
SET SUSI = {ON|OFF}
```

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

where:

```
SET BINS = nn (or calculated by the FOCUS Database Server)
```

Controls the number of I/O buffers for the FOCUS Database Server. Up to 63 bins (shared by all users) are permitted. Set BINS to the maximum for optimal performance. If BINS are not set in the profile, the FOCUS Database Server calculates how many to allocate based on available storage.

Note that source users can issue the SET BINS command for their own processing. When the SET BINS command is included in the FOCUS Database Server profile, it affects only simultaneous usage processing.

```
SET SUSI = {OFF|ON}
```

```
SET SUTABSIZE = {nnn/32}
```

```
SET CACHE={nn|NONE}
```

Cache memory buffers FOCUS database pages between disk and BINS, reducing disk I/O. Issuing SET CACHE in the profile keeps the entire database in memory, thereby improving performance. The default is no cache memory. See your FOCUS documentation for 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 FOCUS Database Server profile, it only affects simultaneous usage processing.

`SET SUSI=ON`

The FOCUS Database Server has the ability to check authorizations through an external security package on MVS. This is activated through this command in the profile.

The default is OFF. The FOCUS SU Security interface is discussed in *Protecting FOCUS Databases: the FOCUS/SU Security Interface (SUSI)* on page 2-16.

`SET SUTABSIZE=nnn`

Enables you to control the number of entries in the user table SU uses to track active users known to the external security system. This value can range from 2 to 256. The default value is 32.

`SET PATHCHECK={ON|OFF}`

Enables you to control whether the FOCUS Database Server 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).

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.

The following error message applies specifically to the SU profile:

**(FOC725) ERROR IN NLI PROFILE**

Selected SET commands may be issued in the HLI profile. This message indicates that 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 data set on MVS). The erroneous line was ignored.

## Stopping the FOCUS Database Server

To shut down the FOCUS Database Server, execute the HLIKX utility found in the partitioned data set FOCLIB.LOAD. From TSO, enter:

```
ALLOC F(FOCSU) DA(dataset) SHR
CALL FOCLIB.LOAD (HLIKX)
```

where:

*dataset*

Names the communication data set that the FOCUS Database Server allocated to ddname FOCSU.

When you execute this program, the FOCUS Database Server processes current transactions, pauses, closes all files, and stops - this usually takes less than a minute.

The FOCUS Database Server can also be stopped with a batch job:

```
//STEP1 EXEC PGM=HLIKX
//STEPLIB DD DSN=FOCLIB.LOAD,DISP=SHR
//FOCSU DD DSN=dataset,DISP=SHR
```

where:

*dataset*

Is the data set name of the communication data set that the FOCUS Database Server allocated to ddname FOCUSU.

Never cancel the FOCUS Database Server. Instead, stop it with the HLIKX utility to ensure proper completion of all pending transactions. FOCUS Database Servers started with passwords can also be stopped by an HLI control command.

## Logging FOCUS Database Server Activity in HLIPRINT

FOCUS can record all user activity on the FOCUS Database Server in a sequential file. Each HLI command is a single action. Each MODIFY/MAINTAIN transaction or TABLE request may execute a series of actions, such as database opens and closes, reads, key matches, and value changes. When you use the ECHO option, the sequential file, ddname HLIPRINT records:

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

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:

- The date and time of the action.
- The amount of CPU time it took to execute the action.
- The number of I/O operations required to execute the action.
- For FOCUS users, the name of the FOCUS procedure (FOCEXEC) executing the action (or a blank field for interactive commands).
- For MODIFY/MAINTAIN requests using Case Logic, the name of the case executing the action or a blank field for non-case logic procedures.

The HLIPRINT file has three uses:

- It provides a guide for restarting procedures that end prematurely. The restart must be tailored to each application, however, as the HLIPRINT file does not log transaction data.
- It provides a means for usage accounting based on the number of actions taken and on the amount of resources consumed.
- It provides a means for monitoring resource utilization, particularly during testing of new procedures.

## HLIPRINT File Contents

Here is a sample of the HLIPRINT file obtained with the ECHO option. A description of the columns follows.

CMD	FILENAME	STATUS	NEWSEG	TARGET	ANCHOR	NTEST	USERID	REF	NUMB
OPN	FOCUSSU	0					WIBDIW	00000001	
RD	CAR	0				1	WIBDIW	00000002	
OPN	CAR	0					WIBEFO	00000003	
MINT	CAR	0					WIBEFO	00000004	
MATA	CAR	0		ORIGIN			WIBEFO	00000005	
MATB	CAR	0		COMP			WIBEFO	00000006	
MDEL	CAR	0		COMP			WIBEFO	00000007	
MATB	CAR	0		ORIGIN			WIBNNR	00000008	
MATB	CAR	1		COMP			WIBNNR	00000009	
MINC	CAR	0		COMP			WIBNNR	00000010	
MATB	CAR	0		ORIGIN			WIBMHK	00000011	
MATB	CAR	0		COMP			WIRMHK	00000012	
MDEL	CAR	0		COMP			WIBMHK	00000013	
MATB	CAR	0		ORIGIN			WIBEFO	00000014	
MATB	CAR	1		COMP			WIBEFO	00000015	
MINC	CAR	0		COMP			WIBEFO	00000016	
MATB	CAR	0		ORIGIN			WIBNNR	00000017	
MATA	CAR	0		COMP			WIBNNR	00000018	
MINC	CAR	1		COMP			WIBNNR	00000019	
CLO	CAR	0					WIBNNR	00000020	
CLO	FOCUSSU	0					WIBNNR	00000021	

### *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/MAINTAIN/SU initialization.

MATA/MATB/MATC/MATD

Match a field value or values in the database (MODIFY/MAINTAIN MATCH subcommand)

**MINC**

Include a new segment instance (MODIFY/MAINTAIN INCLUDE subcommand).

**MUPD**

Update a segment instance (MODIFY/MAINTAIN UPDATE subcommand).

**MDEL**

Delete a segment instance and its descendents (MODIFY/MAINTAIN DELETE subcommand).

**MNEX**

Move to the next segment instance in the segment chain (MODIFY/MAINTAIN TEXT subcommand).

**MREP**

Move to the first segment instance in the segment chain (MODIFY/MAINTAIN REPOSITION subcommand).

**MCMT**

COMMIT subcommand.

**MRBK**

ROLLBACK subcommand.

**SAV**

Write transactions from buffer to the database

**CLO**

Close a file.

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

**FILENAME**

The ddname of the centrally controlled database on which the action took place. Note that the FILENAME column in the Figure above lists a file allocated to the ddname FOCUS<sup>SU</sup>. This is a FOCUS work file.

**STATUS**

Contains either the value of the FOCURRENT variable or the HLI status code for the action. If the action ended normally, this value is set to 0. If a MATCH statement failed to locate a segment instance (ON NOMATCH condition), or if a NEXT statement reached the end of a segment chain (ON NONEXT condition) this value is set to 1. Otherwise, it is the FOCURRENT variable or the HLI status code for the action. Appendix C lists the HLI status codes.

**NEWSEG**

The name of the first segment of new information - for HLI only.

**TARGET**

The segment read or modified by the action.

**ANCHOR**

The anchor segment - for HLI only.

**NTEST**

The number of test conditions qualifying a retrieved record. For FOCUS users, the number of the database page that was read by an RD action. A page is a 4096-byte physical record of a FOCUS database.

**USERID**

The TSO user ID or job ID that issued the action.

**REF NUMB**

The action number in the HLIPRINT file. REF NUMB number is useful with the FOCUS ? FILE command. This query displays a chart showing when each segment in a FOCUS database was last changed (this feature is described fully in the FOCUS Users Manual). The last column in that chart, LAST TRANS NUMBER, lists a number for each segment in the file. If the file was last modified under SU, these numbers 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).

## HLIPRINT File Contents (Extended Format)

The following figure shows a sample extended HLIPRINT file. The columns are the same as those in the HLIPRINT file described in *HLIPRINT File Contents* on page 2-8, with the exception of the additions described. If you display the file on a system editor, note that the first column is reserved for print control characters.

LCMD	FILENAME	STATUS	REUSEG	TARGET	ANCHOR	NTEST	USERID	REF NUMB	DATE	TIME	UTIME	TTIME	IOS	PROC	NAME	CASE	NAME
0																	
OPN	FOCUS3U		0				QCS6AT	00000001	010111	114832	.0000	.0000		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000002	010111	114832	.0137	.0127		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000003	010111	114832	.0009	.0009		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000004	010111	114832	.0006	.0006		1	GT50		
ED	D3OC02		0				1 QCS6AT	00000005	010111	115256	.0009	.0009		1			
ED	D3MF0C02		0				1 QCS6AT	00000006	010111	115210	.0007	.0007		1			
ED	D3MF0C02		0				1 QCS6AT	00000007	010111	115210	.0006	.0006		1			
ED	D3MF0C02		0				1 QCS6AT	00000008	010111	115215	.0008	.0008		1			
ED	D3MF0C02		0				1 QCS6AT	00000009	010111	115405	.0006	.0006		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000010	010111	115405	.0007	.0007		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000011	010111	115405	.0006	.0006		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000012	010111	115421	.0005	.0005		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000013	010111	115421	.0006	.0006		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000014	010111	115421	.0006	.0006		1	GT50		
ED	D3MF0C02		0				1 QCS6AT	00000015	010111	122414	.0024	.0024		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000016	010111	122414	.0006	.0006		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000017	010111	122414	.0007	.0007		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000018	010111	122538	.0012	.0012		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000019	010111	122538	.0006	.0006		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000020	010111	122538	.0007	.0007		1	D3RMO050		
ED	D3MF0C02		0				1 QCS6AT	00000021	010111	124210	.0013	.0013		1	D3RMO051		
ED	D3MF0C02		0				1 QCS6AT	00000022	010111	124210	.0008	.0008		1	D3RMO051		
ED	D3MF0C02		0				1 QCS6AT	00000023	010111	124210	.0006	.0006		1	D3RMO051		
CLO	D3MF0C02		0				QCS6AT *	00000024	010111	125424	.0228	.0228		0	D3RMO051		
OPN	FOCUS3U		0				QCS6AT	00000025	010111	125424	.0352	.0352		1	D3RMO051		

**Sample Extended HLIPRINT File**

Note that a Master File is supplied for the extended HLIPRINT file (member HLIPRINT in MASTER.DATA on the installation tape) enabling you write FOCUS reports against it. See *Producing HLIPRINT Reports* on page 2-13 for details on producing these reports.

**DATE**

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

**TIME**

Time the action finished execution, in HI-IMMSS (hours, minutes, seconds) format.

**VTIME**

Total elapsed CPU time for executing the action (as indicated in the ASCBEJST field of the ASCB for the FOCUS Database Server). 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**

Same as the VTIME column.

**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 on other files such as Master Files. 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. For users entering FOCUS commands 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 their own contents for these words and have them displayed in the PROCNAME column.

**CASE NAME**

For MODIFY/MAINTAIN requests only.

If the request defines a temporary field called SUPRINTNAME (pronounced "SU print name"), the value of this field. SUPRINTNAME must be an alphanumeric field no longer than 12 characters. For example, if your MODIFY/MAINTAIN request contains the statement:

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

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 uses Case Logic, this contains the name of the case that executed the action. Otherwise, CASE NAME is blank.

## *Procedure*

### **How to Create and Use an HLIPRINT File**

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

1. When recording the HLIPRINT file on disk or tape, make sure there is adequate storage space for the file. HLIPRINT files can grow very quickly; a single MODIFY/MAINTAIN transaction or TABLE command can generate several actions and several lines of output, so a day's use can produce thousands of lines. When the space reserved for HLIPRINT is exhausted, the FOCUS Database Server cannot recover.

2. Allocate the HLIPRINT file in the JCL for the sink machine. Give the file the following characteristics:
  - For simple HLIPRINT files (the ECHO option): records are fixed-length and blocked format (RECFM=FB), with a length of 88 characters (LRECL=88) and the block size must be some multiple of 88 (for example, BLKSIZE=8800).
  - For extended HLIPRINT files (the STAT option): records are fixed-length and blocked (RECFM=FB), with a length of 133 characters (LRECL=133). The block size must be a multiple of 133 (for example, BLKSIZE=1330).
  - In general, the larger the block size, the fewer I/O operations FOCUS has to perform. Keep in mind, however, that if the FOCUS Database Server terminates abnormally, all records on the last unwritten block are lost.

Give HLIPRINT the following disposition:

- Use NEW for a new file.
- Use OLD for existing files if you want the new output to replace the present file contents.
- Use MOD for existing files when you want to append new output to the existing file contents.

Here's a sample DD statement for allocating a new file using the simple format:

```
//HLIPRINT DD DSN=SULOG.DATA,  
//          DCB=(LRECL=88,BLKSIZE=8800,RECFM=FB),  
//          DISP=(NEW,CATLG),UNIT=SYSDA
```

This DD statement allocates a new file for the extended HLIPRINT format:

```
//HLIPRINT DD DSN=SULOG.DATA,  
//          DCB=(LRECL=133,BLKSIZE=13300,RECFM=FB),  
//          DISP=(NEW,CATLG),UNIT=SYSDA
```

This DD statement allocates an existing file with the new output appended to the present file contents:

```
//HLIPRINT DD DSN=SULOG.DATA,DISP=MOD,DCB=...
```

The HLIPRINT file can be allocated to tape, to disk, or to a SYSOUT queue.

3. Use the following EXEC statement in the FOCUS Database Server JCL to execute the HLISNK program:

```
//HLISNK EXEC PGM=HLISNK, PARM={'password ECHO'|'password STAT'}
```

where:

ECHO

Records activity in a simple HLIPRINT file.

STAT

Records activity in an extended HLIPRINT file.

The password is optional. If you choose to omit it, remember to type a leading blank before the ECHO or STAT keyword. Otherwise, FOCUS will not interpret it correctly.

## Producing HLIPRINT Reports

The extended HLIPRINT trace facility has an associated Master File called HLIPRINT MASTER. This enables you to produce summary reports of your HLIPRINT trace information using FOCUS TABLE commands. These summary reports can be used to:

- Determine peak FOCUS Database Server usage by user, time of day, database, or application.
- Troubleshoot performance problems in procedures or database design.
- Perform usage accounting.

Before you can issue TABLE commands against HLIPRINT, you must allocate ddname HLIPRINT to the trace file. If you named your HLIPRINT data set MYTRACE.HLIPRINT.DATA, you would enter the following command from within FOCUS:

```
TSO ALLOCATE F(HLIPRINT) DA('MYTRACE.HLIPRINT.DATA') SHR
```

You must also allocate the ddname MASTER to the Master File PDS supplied within FOCUS.

After allocating HLIPRINT and MASTER, you can issue TABLE requests against the HLIPRINT file.

The following sample request determines how much of the sink machine's resources were used by each userid:

```
TABLE FILE HLIPRINT  
SUM CPU AND IOS  
BY USERID  
END
```

This request produces a report showing the total elapsed job step time and the number of I/O operations used by the sink machine on behalf of each userid.

The next sample request can be used to locate performance problems in an application:

```
TABLE FILE HLIPRINT SUM CPU AND IOS
BY PROC_NAME BY CASE_NAME ON PROC_NAME SUMMARIZE
END
```

Here is the resulting report:

PROC_NAME	CASE_NAME	CPU	IOS
CARLOAD		.3204	6
*TOTAL CARLOAD		.3204	6
CARMOD	ADDCASE	1.1040	17
	DELCASE	.5919	10
	LOCATE	19.8299	428
	SHOWCASE	1.2534	5
	TOP	.5745	6
	UPDATECASE	1.1790	21
*TOTAL CARLOAD		24.5327	487
TOTAL		24.8531	493

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

For more information about report requests, see your FOCUS documentation.

## Calculating Core Requirements for the FOCUS Database Server

When ten or fewer users are working with two databases each of which has several hundred fields, you need about a megabyte of storage to run the FOCUS Database Server. Generally speaking, the core needed to run the FOCUS Database Server equals the sum of the size of the FOCUS Database Server code (620K) plus operating system overhead (approximately 250K), plus overhead for the database and the user(s).

Operating system overhead is approximately 250K, but depends on the size of the directory and on the number and size of the files being opened.

### Procedure

### How to Calculate Database Overhead for Centrally Controlled Databases

To calculate database overhead, compute the database overhead for each centrally controlled database processed by the FOCUS Database Server using the following formula:

$$(s * 48) + (f * 72) + 200$$

where:

*s*

Is the number of segments in the database.

*f*

Is the number of fields in the database.

Only HLI and MODIFY/MAINTAIN create user overhead. Calculate user overhead with the following steps:

1. Estimate the maximum number of users who might use the FOCUS Database Server simultaneously.
2. For each database, sum the field lengths as defined by the FORMAT attribute of the Master Files. Integer and floating-point fields are 4-bytes long and packed and double-precision fields are eight bytes. Include in this sum the lengths of cross-reference fields (that is, fields segment types of KM, KU, KL, and KLU). *Do not* include lengths of fields in databases that you intend to join to by using the JOIN command.
3. HLI usage affects computation of user overhead:
  - If no one is using HLI, take the maximum total field length and add 3200. Then multiply that by the number of users. The result is the user overhead.
  - If someone is using HLI, add the total field lengths for all files that could be opened concurrently by a single user, plus 3200 bytes per file, plus four bytes for each field specified by the HLI SHO command. Then multiply the sum by the number of users to obtain total user overhead.

## Example

### Calculating Database Overhead and Total Field Length

Consider the following Master File:

```
FILE=CARDATA, SUFFIX=FOC,$
SEGNAME=CARSEG, SEGTYPE=S1,$
FIELDNAME=CAR, ALIAS=AUTO, FORMAT=A16,$
FIELDNAME=MANUFACTURER, ALIAS=MAKE, FORMAT=A20,$
SEGNAME=MODELSEG, SEGTYPE=S1, PARENT=CARSEG,$
FIELDNAME=MODEL, ALIAS=MODEL, FORMAT=A22,$
FIELDNAME=SEATS, ALIAS=SEAT, FORMAT=I4,$
FIELDNAME=DEALER_COST, ALIAS=DCOST, FORMAT=D7,$
FIELDNAME=RETAIL_COST, ALIAS=RCOST, FORMAT=D7,$
FIELDNAME=SALES, ALIAS=UNITS, FORMAT=I6,$
```

This database has two segments and seven fields. The database overhead is  $(2 \times 48) + (7 \times 72) + 200 = 800$  bytes. The total field length is:

$16 + 20 + 22 + 4 + 8 + 8 + 4 = 82$  bytes

The next example shows how to calculate minimum core requirements for a FOCUS Database Server. Assume a site has 20 terminals available for its data entry staff. The staff uses two databases:

- The first has a database overhead of 1800 bytes and a total field length of 450 bytes.
- A second has a database overhead of 1700 bytes and a total field length of 750 bytes.

The FOCUS Database Server code takes 620K. The system overhead is 250K. The total database overhead is 3500 bytes or roughly 4K.

**Example**

**Calculating Core Requirements When HLI Is Not in Use**

If no one is using HLI, choose the database with the larger total field lengths (750 bytes for the second database). Adding 3200 bytes to this number gives 3950 bytes, or roughly 4K. Since 20 is the maximum number of data entry clerks that can be on a FOCUS Database Server at the same time, the user overhead is:

$$4\text{K} \times 20 = 80\text{K}$$

Since Database Server core requirements equal the sum of the Database Server code size, plus system overhead, plus database overhead, plus user overhead, the minimum amount of core required is:

$$620\text{K} + 250\text{K} + 4\text{K} + 80\text{K} = 954\text{K}$$

**Example**

**Calculating Core Requirements When HLI Is in Use**

If someone were using HLI, you would add the total field lengths of the two databases (450 + 750), which gives us 1200 bytes. Since there are two databases, add 2 x 3200 or 6400 bytes. Suppose that there were five fields of four bytes each listed in any SHO command. Their contribution would be 20 bytes. Adding 1200 plus 6400 plus 20 yields 7620 bytes. Since no more than 20 data entry clerks could be on the Database Server at a time, the user overhead would be:

$$7620 \times 20 = 152,040 \text{ bytes, or about } 152\text{K}$$

The Database Server needs the sum of the Server code size, plus system overhead, database overhead, and user overhead, so the minimum amount of core needed is:

$$620\text{K} + 250\text{K} + 4\text{K} + 152\text{K} = 1026\text{K}$$

Appendix A provides suggestions for improving SU performance.

## Protecting FOCUS Databases: the FOCUS/SU Security Interface (SUSI)

The FOCUS/SU Security Interface ensures that FOCUS databases controlled by FOCUS Database Servers will be properly protected by third-party security packages such as RACF, ACF2, or TOP SECRET. The Interface uses the external security package to check authorization for specific source user IDs and specific data sets, as if the source user ID requested the data set locally.

The FOCUS/SU Security Interface augments existing FOCUS DBA security for FOCUS applications; it does not affect local users or non-SU files.

The FOCUS/SU Security Interface is included on the FOCUS distribution tape. For installation instructions, see the *OS/390 and MVS Installation Guide*.

## Using the SU Security Interface

To use the FOCUS/SU interface, insert the following command in the SU profile:

```
SET SUSI=ON
```

The SU profile is member HLIPROF of a PDS allocated to ddname FOCEXEC on the sink machine.

Once the Interface is installed, you must change all FOCUS Database Server jobs to execute the program HLISECUR, rather than HLISNK, as in the following JCL line:

```
//SINK EXEC PGM=HLISECUR,PARM='parameters'
```

You can use the same password and ECHO or STAT parameters for HLISECUR as for HLISNK.

## How the FOCUS Database Server Invokes the Security Interface

The FOCUS Database Server invokes the FOCUS/SU Security interface (SUSI) when a user executes a request against a shared database on the server. When a user opens a database, the sink machine asks the security system if the user is authorized to read and/or write to that database.

For MODIFY/MAINTAIN requests, users must have both read and write access. Authorization is checked twice: first for read access and then for write access. Both authorization checks are done for all MODIFY/MAINTAIN requests, including read-only MODIFY/MAINTAIN requests. Cross-referenced files are also checked for read access.

On TABLE requests, a read-access authorization check is done each time FOCUS reads page 1 of a database. This includes and cross-referenced files and files linked with JOIN. HLI/SU users are checked for read and write authorization at the time that the OPN command is issued.

## How the FOCUS Database Server Checks Access Rules

The FOCUS/SU Security Interface accesses MVS security packages via the RACINIT and RACHECK variants of the RACROUTE macro. These functions trigger security routine execution via the System Authorization Facility (SAF), rather than by native RACF. Since security packages such as RACF, ACF2, and TOP SECRET all interface to SAF in the same way, the FOCUS/SU Security Interface operates transparently with any of these packages.

The FOCUS Database Server may discover data sets that are not protected by the external security system (that is, no specific security rule has been written for that dataset). In this case, the database Server next checks to determine if the source user has security access to the high-level qualifier of the data set. If no rule was written for the high-level qualifier, the Database Server will reject the source user's request.

In addition to observing the rules protecting databases from source user access, the FOCUS Database Server must itself be authorized to open the databases under its control.

## How the FOCUS Database Server Manages the Number of Users

When users first access database controlled by the FOCUS Database Server, the Server issues an authorization request to the external security package as if it were the source user. It then keeps a list of active users known to the external security system in a table that has a fixed number of slots. The default is 32 users. The user ID stays in the list until the last user has logged off.

When the user table is filled, the sink removes the oldest user from the list and adds a new user. This enables more than 32 users to use the Server concurrently, but only the most recent 32 remain on the external security system's active list at one time. This benefits those making frequent requests against Server databases. If a user was removed from the list (because 32 other users made more recent requests), a subsequent request from the inactive user causes the FOCUS Database Server to issue a new authorization request to the external security package.

Since each authorized user slot requires approximately 256 bytes of common storage area (CSA), you may choose to limit the number of simultaneously active users. You can control the number of entries in the user table through the command SET SUTABSIZE.

### Syntax

#### Establishing the Maximum Number of Simultaneously Active Users

```
SET SUTABSIZE={nnn | 32}
```

where

*nnn*

Is any number of entries from 2 to 256. The default value is 32.

Place this command in the profile for the FOCUS Database Server, which resides in member HLIPROF in a FOCEXEC PDS allocated by the Database Server job.

Note that the use of this command does not affect the number of users who can simultaneously use the sink machine. SUTABSIZE simply provides a way to put a ceiling on the amount of CSA used by the Interface.

If you increase the size of the user table above the default size of 32, CSA utilization will increase by approximately 256 bytes for each slot being used. Note that the 256 bytes of CSA per user is only used if a user occupies that slot in the table. If the value of SUTABSIZE is larger than the number of active users, the amount of CSA used is not affected by changing the setting of SUTABSIZE.

## Error Messages

The following error will occur when a source user ID attempts to access a file to which it does not have access:

**(FOC517) SU.ACCESS DENIED BY EXTERNAL SECURITY SYSTEM:**

This message indicates that the external security system (ACF2, RACF, or TOP SECRET) determined you have no access rights to a file controlled by the FOCUS Database Server.

If the FOCUS Database Server was started with the ECHO or STAT parameter a status code of 777 appears in HLIPRINT. HLI/SU users get a status code of 777 in word 24 of the File Communications Block WC13) when they issue an OPN command for a file for which they do not have access.

Rejected security requests are all logged in the FOCUS Database Server job output stream. In addition, rejected SU security requests are also logged on the operating system operator console, as with other attempted operating system security violations. Return and reason codes also appear in the error log on ddname HLIERROR (or HLIPRINT if HLIERROR is not allocated), but for security, they are not displayed at the user's terminal.

If the exit is activated by placing SET SUSI=ON in the SU profile but the FOCUS Database Server is started by invoking HLISNK instead of HLISECUR, a 047 System ABEND occurs when the first user accesses the Database Server.

If HLISECUR were invoked but the exit was not activated as described above, no indication would be given and the FOCUS Database Server would provide no access checking.

---

## CHAPTER 3

# SU and The FOCUS Language

### Topics:

- Using Centrally Controlled Data Sources
- Using the Multi-threaded SU Reporting Facility
- Messages From the FOCUS Database Server
- Using MODIFY or Maintain With SU

This chapter shows how to read and update centrally controlled data sources using FOCUS.

SU users can issue all FOCUS commands, with the following restrictions:

- The SCAN and FSCAN facilities are read/only.
- When using COMBINE for combining several data sources, all must be controlled by the same FOCUS Database Server.
- End users cannot issue REBUILD or RESTRICT commands or issue CREATE commands for data sources controlled by the FOCUS Database Server.

## Using Centrally Controlled Data Sources

When using centrally controlled data sources you communicate through the FOCUS Database Server that controls them. So, before entering SU, you must first allocate a special communication data set for each FOCUS Database Server you will use. For each allocation supply the following parameters:

- A ddname of your choice.
- The data set name of the file allocated to ddname FOCSU by the sink machine
- A disposition of SHR.

Assume, for example, that you will use two FOCUS Database Servers. One server has communications data set SYS1.SU01.DATA allocated to ddname FOCSU and the other has communication data set SYS1.SU02.DATA allocated to ddname FOCSU. To allocate these data sets on your TSO user ID, you would issue the following ALLOC commands:

```
ALLOC F(SYNCA) DA('SYS1.SU01.DATA') SHR
ALLOC F(SYNCB) DA('SYS2.SU02.DATA') SHR
```

You must allocate them in SHR mode so that others can also allocate them. You will reference individual servers by the ddnames to which their communication data sets are allocated. In the above example, the first server is SYNCA and the second is SYNCB.

Unless you are running the Multi-Threaded SU Reporting Facility, you do not allocate centrally controlled databases. Instead, you allocate the files containing the Master Files for these data sources, which must be identical to the descriptions contained in the files allocated to the FOCUS Database Server.

Allocate all other files as if you were using FOCUS locally.

### Syntax

### How to Specify FOCUS Data Sources for Use in SU Mode

```
USE
.
.
file1 ON communication
file2 ON communication
.
.
END
```

where:

*filen* ...

Is the ddname of a data source.

*communication*

Is the ddname of the communication data set you allocated on your ID. This data set must be allocated to ddname FOCSU on the FOCUS Database Server.

Suppose, for example, you wish to use the EMPLOYEE, EDUCFILE, and JOBFIL data sources with SU. In our example, the FOCUS Database Server with data set ddname SYNCA controls the EMPLOYEE and EDUCFILE data sources, and the FOCUS Database Server with data set ddname SYNCB controls the JOBFIL data source.

You would therefore enter the following USE command:

```
USE
EMPLOYEE ON SYNCA
EDUCFILE ON SYNCA
JOBFILE ON SYNCA
END
```

You could also use this same USE command to designate any other data sources you plan to use locally.

The ON option with AS in USE commands is supported, provided the keyword READ is included:

```
USE fileid AS mastername ON userid READ
```

**Note:** The READ option does not control whether a user has read or write access to the data source, it is required because alternate Master Files are not supported on SU for MODIFY or Maintain.

## Using the Multi-threaded SU Reporting Facility

The Multi-Threaded SU Reporting facility enables you to issue TABLE requests against shared data sources without using a FOCUS Database Server. Before using it, you must first allocate the centrally controlled data source in SHR mode, as shown below:

```
ALLOC F(CAR) DA('SINKID.CAR.FOCUS') SHR
```

This is the only case in which a source machine can allocate the centrally controlled database.

This feature may only be used in read/only mode and all modifications to the data source must go through the FOCUS Database Server. In addition, a single Multi-Threaded HLI/SU program cannot simultaneously open a file in both read/only and read/write modes. Separate FCBs are required for locally reporting from and updating centrally controlled data sources.

In OS/390 and MVS, when performing Multi-Threaded HLI/SU Reporting operations, the central database is accessed locally, bypassing the FOCUS Database Server. But for all modifications, central data sources are accessed through the FOCUS Database Server. In the HLI environment, the FCB is modified to denote the parallel configuration (accomplished in the FOCUS environment with a USE command).

The steps for allocating the database are as follows (these assume that the FOCUS Database Server is running on user ID SINKMA):

1. Allocate the database in the sink job using DISP=SHR.

For example:

```
//CAR DD DSN=SINKMA.CAR.FOCUS,DISP=SHR
```

2. Allocate the database for the HLI program using DISP=SHR. For example:

```
ALLOC F(CAR) DA('SINKMA.CAR.FOCUS') SHR
//CAR DD DSN=SINKMA.CAR.FOCUS,DISP=SHR
```

3. Set up the FCB per the following steps:
  - a. Set FCB word 6 (SU) to SULO.
  - b. Set FCB word 9 (SINKID) to the ddname of the FOCUS communication dataset.

For example, assume FOCSU is the communication dataset 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:

```
IOS-SET-UP-FCB-HLI-SU-MVS.
      MOVE "CAR"      TO FCB_FN.
      MOVE "SULO"     TO FCB_SU.
      MOVE "FOCSU"    TO MB_SINKID.
```

The USE command is issued differently for multi-threaded reporting. The following example illustrates that each database you want to use must be declared twice:

```
USE
CAR ON SINKID
CAR LOCAL
ANOTHER ON SINKID2
ANOTHER LOCAL
.
.
END
```

In the example, the two FOCUS databases, CAR and ANOTHER reside on different FOCUS Database Servers. The keyword LOCAL indicates that the Multi-Threaded SU Reporting Facility will access them without going through their associated FOCUS Database Servers for TABLE requests, while MODIFY/Maintain requests will go through the FOCUS Database Servers as usual.

Note that you must still provide a USE command to tell FOCUS what FOCUS Database Server controls the database, even when planning to read the data source locally with the Multi-Threaded SU Reporting Facility.

## Special Considerations for Using Multi-threaded SU Reporting

While reading data with this facility, you will not know if someone else is changing the data source with a MODIFY or Maintain request. During a TABLE request, if FOCUS detects that a record it is retrieving has been affected by another user's changes, it re-requests the changed record from the FOCUS Database Server. This is why you must declare the FOCUS Database Server in the USE command. Since reports you generate could be affected by other users' changes, keep the following in mind:

- To find out if someone else is using your data source via SU, issue the ? SU query. If no MODIFY or Maintain users are on the data source when your report is generated, the data will not change.
- If another user is modifying records while your TABLE request is executing, your results could contain a combination of modified and unmodified data.
- If a root segment instance (or root segment of an alternate view) is deleted and reinserted while your report is in progress, the segment may show up twice.

## Messages From the FOCUS Database Server

This section covers messages generated by the FOCUS Database server, which include:

- Error messages.
- Responses to the ? SU query.

### Error Messages

When the FOCUS Database Server has a problem while processing your request, FOCUS sends back an error message. The most common problem is that the FOCUS Database Server is not active because:

- You did not specify the correct communication data set ddname in your USE command.
- The FOCUS Database Server was never started.
- The FOCUS Database Server was not given enough memory.
- The FOCUS Database Server was shut down.

If you send the FOCUS Database Server a request or transaction when it is not operating, you will receive one of the following error messages (note that the FOCUS Database Server was originally called the sink machine):

- FOC542 - if you specified the wrong communication data set in your USE command or never started the FOCUS Database Server.
- FOC543 - if you did not give the FOCUS Database Server enough memory when you started it.
- FOC544 - if the FOCUS Database Server was shut down.

If the FOCUS Database Server stops normally while you are executing a MODIFY or Maintain request, all of your transactions are written to the database. If it stops abnormally, some or all of your last transactions could be lost. To minimize such losses, see *The Checkpoint Facility* on page 3-13.

If you use an alternate Master File via the FOCUS Database Server and do not include the READ keyword in the USE list, the following error message is displayed:

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

## Listing FOCUS Database Server Users

When preparing to use (or using) a FOCUS Database Server, you can see who else is logged on and the ddnames of the databases they are using by issuing the following command at the FOCUS command level:

```
? SU ddname
```

where

ddname

Is the ddname you allocated to the communication data set for the FOCUS Database Server. This is a convenient way for SU administrators to monitor user activities.

### Example

#### Inquiring About Active Users on a FOCUS Database Server

If user WIBMLH wishes to inquire about the FOCUS Database Server, for which he allocated the communication data set named SYNCA, he issues the following:

```
? SU SYNCA
```

which might return the following listing:

```
USERID FILEID QUEUE
```

```
WIBMLH QUERY
```

```
WIBJBP CAR
```

This output shows that WIBMLH made the request and that user ID WIBJBP is now using the CAR database on that Server. If there were no database users on the Server, FOCUS would return the following message:

```
NO SIMULTANEOUS USERS ARE ACTIVE ON SYNCA
```

If SYNCA were not operating, one of the error messages previously described would appear (see *Error Messages* on page 3-5).

## Status Testing with ? SU Queries

You can use ? SU queries to test return code (RETCODE) values in FOCEXEC procedures. When a ? SU command is executed, the query results determine the value of the Dialogue Manager status return variable. The query results and the corresponding RETCODE values are shown in the table below.

Query Result	RETCODE Value
Users are active on sink machine (FOCUS Database Server)	0
No users are active and the sink machine is running	8
Sink machine is not running	16

The RETCODE value is later tested in an IF command. For example, the following procedure could be used to test for FOCUS Database Server availability before initiating a MODIFY or Maintain procedure.

```
? SU ddname
-RUN
-IF &RETCODE GT 8 GOTO BAD
-INCLUDE myproc
-BAD
-EXIT
```

where:

*ddname*

Is the ddname allocated to the communication data set of the FOCUS Database Server.

*myproc*

Is the name of your MODIFY or Maintain procedure

If RETCODE has a value of 0 or 8, the specified procedure will be executed.

## Using MODIFY or Maintain With SU

This section discusses conflict resolution when using MODIFY or Maintain against centrally controlled data sources. In particular, it discusses the FOCURRENT variable. The section covers:

- The FOCURRENT variable, which indicates if your transaction is rejected because of a conflict. **Note:** In Maintain, the change verification variable is called FocCurrent. The remainder of this discussion uses the term FOCURRENT to mean FOCURRENT in MODIFY or FocCurrent in Maintain.
- Use of the FOCURRENT variable in testing for conflicts.
- The FOCURRENT variable and CRTFORM turnaround fields.

In Maintain you must issue the following command before running the application:

```
SET COMMIT = ON
```

**Tip:** Limit the number of records retrieved into a stack to reduce the chances of a conflict.

## Evaluating Conflicts: The FOCURRENT Variable

When you submit a MODIFY/Maintain transaction under SU, FOCUS stores information about it in the FOCURRENT variable. Its value indicates whether or not there is (or could be) a conflict with another transaction. A zero (0) value indicates there is no conflict and the transaction is accepted. A non-zero value indicates apparent conflict, so the transaction is rejected and an error message is issued.

Note that FOCUS treats rejections due to conflicts between transactions just like record-validation failures. You can log these invalid transactions using the LOG INVALID command described in your FOCUS documentation.

When designing MODIFY or Maintain requests, you can test FOCURRENT and branch according to the results. For example, a MODIFY/Maintain request can submit a transaction, test FOCURRENT, and if its value is not 0, resubmit the transaction. This is explained in Testing for Rejected Transactions on page 3-9.

The values assigned FOCURRENT are:

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

The following chart shows possible values for FOCURRENT following different types of transactions. The rows list types of transactions:

User-desired action	No simultaneous action	Instance simultaneously updated	Instance simultaneously deleted	Instance simultaneously included
UPDATE	0	3	2	-
DELETE	0	3	2	-
INCLUDE	0	-	-	1

In summary, FOCURRENT will not contain a zero (0) if:

- You cannot update or delete a segment instance because the record has been changed or deleted by another user.
- You cannot include a new instance because another user already added it.

- One special MATCH case exists in which FOCURRENT is not set to zero. If a MODIFY user issues a MATCH for a particular set of key values, the FOCUS Database Server saves the values of the retrieved segment instance. If the user issues another MATCH on the same key values with no intervening MATCH or NEXT against the database, the server compares the newly retrieved segment instance with the copy it saved on the first MATCH. If these two segment instances are the same, FOCURRENT will be set to zero. If they are different, another user must have changed the segment instance since the original MATCH was performed and FOCURRENT is therefore set to 1. This is the only case in which FOCURRENT is set to a non-zero value on a MATCH command.

An example of this particular situation appears in *Validating CRTFORM Turnaround Fields* on page 3-10.

## Testing for Rejected Transactions

By testing the FOCURRENT variable, MODIFY/Maintain requests can process transactions even after they have been rejected due to conflicts. Such requests are designed using Case Logic, which is discussed in your FOCUS documentation.

For example, assume a MODIFY/Maintain request resubmits a rejected transaction. There are two possible results:

- If the transaction was previously rejected because the target database instance was already updated, this transaction might be accepted a 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 re-submission, result in a NOMATCH condition:

```
MODIFY FILE EMPLOYEE
PROMPT EMP_ID
GOTO NEWSAL

CASE NEWSAL
MATCH EMP-W
  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 for an employee ID and branches to the NEWSAL case. If the ID is in the database, the NEWSAL case prompts 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 for the next employee 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 again for the salary and resubmits the transaction. But if the instance was deleted, the request returns a NOMATCH condition (ON NOMATCH REJECT) and prompts for the next transaction.

## Validating CRTFORM Turnaround Fields

Always include FOCURRENT tests in MODIFY requests that validate CRTFORM turnaround fields. Otherwise, after a transaction is rejected as invalid, you may update a field without knowing that it was already updated.

Turnaround fields are fields with a prefix (T.fieldname) that display database values. They appear in CRTFORMs in MODIFY update requests (for example, T.SALARY). When you execute such a request, the present database value for the turnaround field named (SALARY, in this case) is displayed in the CRTFORM where you can change it or accept it as shown. (For complete information about turnaround fields, see your FOCUS documentation.)

You can set up validation tests on turnaround fields, so that if you attempt to enter an invalid value for a field, the request can reject it and retrieve a fresh copy of the segment instance from the database, while redisplaying turnaround field values on the screen as you entered them (after you press the Enter key a second time).

Since the request redisplay your values from the first attempt, it may not be clear whether someone else updated the 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 the first entries failed the validation tests (this copy contains the values updated by the other user, but these are not displayed).

Therefore, your MODIFY requests should test the FOCURRENT variable after retrieving data from the database with the MATCH command. If you previously matched on the same key values, and the instance currently in the database is not the same as your version, then FOCURRENT will contain a value of 1.

**Example****Testing FOCURRENT to Validate Turnaround Fields**

The following example shows why the FOCURRENT test is needed and how it should be used. This request updates employees' salaries, allowing employees a maximum salary of \$50,000 yearly:

```

MODIFY FILE EMPLOYEE
CRTFORM
  "ENTER EMPLOYEE'S ID: <EMP_ID>"
MATCH EMP_ID
  OM MOMATCH 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 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, some other 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

```

A typing mistake inadvertently enters a salary of \$55,000, exceeding the \$50,000 maximum, so the system returns a message indicating an invalid salary value. You press Enter to redisplay the turnaround fields. FOCUS responds:

```

ENTER EMPLOYEE'S ID: 071382660
ENTER SALARY: 55000.00

```

You see the SALARY field displays 50000, not the updated value 13,000.00, even though the request just retrieved a fresh copy of the instance from the database, so 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 have been correct.

To solve this problem, test the FOCURRENT variable whenever you retrieve copies of instances from the database. You could write the previous MODIFY request in the following manner (additions are bolded):

```
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 FOCURREMT NE 0 GOTO ERROR;
CASE ERROR
TYPE
  "FOCURRENT TEST FAILED"
  "ANOTHER USER CHANGED THIS SEGMENT INSTANCE"
GOTO TOP
ENDCASE
DATA VIA FIDEL
END
```

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 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, it tests the FOCURRENT variable. If another user changed the value of SALARY after you retrieved it, the MATCH on EMP\_ID will return a FOCURRENT value of 1. If FOCURRENT is not zero, FOCUS informs you that your transaction 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 updated SALARY (13000.00) for you to accept or change.

## Protecting FOCUS Database Server Transactions

Two ways exist of protecting pending transactions in cases where system failures bring down the FOCUS Database Server:

- The Checkpoint facility (MODIFY only).
- The subcommands COMMIT and ROLLBACK.

## The Checkpoint Facility

When FOCUS accepts transactions, it writes them to a buffer rather than immediately writing them to the database. When the buffer is full, FOCUS writes all of them to the database at the same time. This reduces the input/output operations required. Should your MVS system fail, however, and bring down the FOCUS Database Server then any transactions in that buffer could be lost. There are several ways to guard against this potential loss of data. First, you can force FOCUS to update the database more frequently by using the Checkpoint facility, which is described in your FOCUS documentation.

Checkpoint is activated with a MODIFY CHECK command, which specifies the number of transactions for FOCUS to accumulate in the buffer before writing them to the database. The following request writes transactions to the database in groups of ten:

```
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 FOCUS Database Server, it automatically writes pending transactions for all users on that database server. It then records this as a SAV action in the HLIPRINT file. FOCUS also writes pending transactions from all users on the same database server when each (user's) MODIFY request finishes execution (a CLO Command in the HLIPRINT file).

Should the system fail while executing MODIFY requests, all transactions entered before FOCUS last wrote to the database are saved in the database. All transactions entered after that may be lost. Using CHECK 1 avoids any lost transactions.

Sometimes it's a good idea to log all transactions in a sequential file to simplify reentry if any are lost. To do this, simply add LOG or TYPE commands to the MODIFY request. LOG commands are satisfactory for simple requests; TYPE commands are better for complex Case Logic requests. For more information, see your FOCUS documentation.

You can also protect the integrity of the centrally controlled database itself with the Absolute File Integrity feature described in your FOCUS documentation.

## Managing Transactions: COMMIT and ROLLBACK

COMMIT and ROLLBACK subcommands provide control over pending database changes (COMMIT) and enable you to undo changes before they are made permanent (ROLLBACK).

COMMIT safeguards transactions in cases of system failures and provides greater control than the Checkpoint facility over which transactions are written to the database.

The MODIFY CHECK command only provides control of the number of transactions required before changes will be written to the database. With CHECK you cannot change the checkpoint setting once the request begins execution or cancel the changes.

COMMIT enables you to make changes based on the content of the transactions as well as the number. Unwanted changes can be canceled with ROLLBACK, unless a COMMIT was issued for them. 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. Absolute File Integrity for databases in SU is provided solely by the FOCUS Shadow Writing Facility. See your FOCUS documentation for information on Absolute File Integrity and the SET SHADOW command.

## COMMIT and ROLLBACK Subcommands

COMMIT and ROLLBACK each process a logical transaction. In the MODIFY or Maintain environments you can group database changes and treat them as one logical update transaction. For example, you can handle multiple records displayed on a CRTFORM and processed via the REPEAT command as a single transaction. The logical transaction is terminated by either a COMMIT or ROLLBACK operation. You can also use COMMIT and ROLLBACK for single-record processing.

When COMMIT ends a logical transaction, it writes all pending changes to the database. Such changes, once committed, cannot be rolled back. COMMIT can be coded as a global subcommand or as part of MATCH or NEXT logic. MATCH and NEXT command possibilities are as follows:

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

When ROLLBACK terminates a logical transaction, it does not write the changes to the database. It simply cancels all pending changes made since the last COMMIT. ROLLBACK cannot cancel changes already committed.

ROLLBACK can be coded as a global subcommand or as part of MATCH or NEXT logic. MATCH and NEXT command possibilities are as follows:

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

Should a COMMIT fail for any reason (for example, system failure, lack of disk space), no changes are made to the database. COMMIT is thus an all-or-nothing feature that ensures database integrity.

## Using COMMIT and ROLLBACK With the FOCUS Database Server

The SU change/verify protocol operates on a logical transaction basis that can encompass the processing of blocks of records. It is based on the optimistic assumption that two users will rarely change the same record at the same time. In SU processing with COMMIT, the FOCUS Database Server keeps a table of soft locks to track the records requested by each user.

When you issue a COMMIT, the database server checks this table for each record you wish to change, to determine if another user may have changed it between the times when you retrieved it and when you issued the COMMIT. If another user has issued a COMMIT against a record that you intended to update, your changes are automatically rolled back.

For example, User A and User B execute a MODIFY/Maintain procedure against the EMPLOYEE database on a FOCUS Database Server. 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 interdependent transactions. For these applications, processing is successful only when several physical transactions are complete. The following application transfers funds from one bank account to another. There are two components to 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 incomplete, no changes should be made to the database. Because the application has multiple users, it is essential that each account be involved in only one transfer transaction at a time.

The BANK file in this application is a FOCUS data source on a FOCUS Database Server with the following Master File:

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

This procedure shows the use of COMMIT and ROLLBACK for ensuring that each logical transaction is complete before it is written to the database. It also illustrates the use of FOCURRENT in SU COMMIT processing. (Numbers to the left of the command lines are keys to annotations following the example.)

```
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.   ON NOMATCH ROLLBACK
  ON NOMATCH REJECT
  ON MATCH COMPUTE BALANCE = D.BALANCE - AMOUNT;
4.   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
```

**Notes:**

1. This 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 pending changes already 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 pending changes already made are rolled back. This is particularly important in CASE ADD, since funds may already have been subtracted (through the UPDATE in CASE SUBTRACT).
7. The COMMIT is executed if both updates are successful.
8. A test for a zero value of FOCURRENT follows the COMMIT command. If the COMMIT fails (if FOCURRENT has a non-zero value) the transaction will be rolled back automatically.

## Reference

### Restrictions on COMMIT and ROLLBACK in SU

Note the following restrictions concerning COMMIT and ROLLBACK:

- COMMIT and ROLLBACK are only supported with FOCUS data sources and will be ignored if included in procedures against non-FOCUS data sources.
- All files referenced by MODIFY/Maintain procedures that use COMMIT and ROLLBACK processing must be specified in USE commands. This includes cross-referenced files and joined files.
- After a COMMIT or ROLLBACK operation against a central database, the procedure no longer has a current position in the file. You must reestablish the current position through a MATCH or NEXT command.
- COMMIT and ROLLBACK are not available for HLI.
- COMMIT and ROLLBACK do not support use of alternate views. For example, the syntax MODIFY/Maintain 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 buffer size used for sending COMMIT transactions to FOCUS Database server machine is 32K bytes.

## SU Performance Considerations

COMMIT and ROLLBACK improve SU performance by providing the ability to group individual transactions into one logical transaction, which takes fewer disk I/O operations. Reducing the number of transactions also minimizes communications between the FOCUS Database Server and the source user IDs. Specifically, a single communication occurs when transactions are rolled back and none is needed to match a record a second time. Both of these situations are common in multi-record processing.

## Comparing COMMIT and ROLLBACK to the Checkpoint (CHECK) Facility

With CHECK processing in MODIFY procedures, the FOCUS Database Server buffers all changes to BINS before saving them to the database. The changes are saved from BINS to disk when a checkpoint is issued through a CHECK command. For example, CHECK 5 tells the procedure to issue a checkpoint after every five (5) database updates.

There are several limitations with the CHECK facility. You have little control over when checkpoints are done. When *any user* on a FOCUS Database Server issues a checkpoint, the server saves pending changes from all other users. More importantly, once committed, those changes cannot be undone. The CHECK number is constant.

COMMIT and ROLLBACK provide a number of advantages over the CHECK facility. The source machine holds all pending changes until a COMMIT is issued and then the changes are sent to the database server. No one else sees your changes until you issue a COMMIT and pending changes can always be canceled with the ROLLBACK command.

COMMIT and ROLLBACK are controlled by the MODIFY/Maintain procedure and are not subject to other users' checkpoints. All or nothing processing is guaranteed, even upon system failure and, since all the changes are sent to the FOCUS Database in one logical transaction, processing is more efficient.

## Using FOCURRENT

The FOCURRENT variable (FocCurrent in Maintain) is intended for use with COMMIT and ROLLBACK and is used for determining whether or not transactions have been accepted. In applications without COMMIT and ROLLBACK, FOCURRENT cannot deal with multiple changes and may be difficult to code.

When using COMMIT and ROLLBACK, you must test FOCURRENT after each COMMIT. If a COMMIT fails, the FOCURRENT value will be set to a non-zero value. MODIFY/Maintain only checks FOCURRENT after a COMMIT, not after every transaction. If the FOCURRENT value is non-zero, this tells MODIFY/Maintain that the entire logical transaction was rolled back.

Note also that since no message is issued when FOCURRENT has a non-zero value, you must test it explicitly after each COMMIT and react appropriately; otherwise there will be no message and processing will return to the top case.

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

Other users' changes are only visible after a COMMIT has been issued and successfully processed. If you issue a MATCH or NEXT on a record more than once and do not issue a COMMIT, the MODIFY or Maintain does not go to the database server again, so you will not see changes made by other users until your procedure executes a COMMIT command, although you do see your own. Once a COMMIT is performed, the record comes from the database when you next issue a MATCH or NEXT on it.

This is an important concept in using the FIND or LOOKUP functions. These operations cause records to be retrieved from the FOCUS Database Server. FIND and LOOKUP do not search for values on 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 ROLLOACK

Two commands are provided in the SU profile to enable FOCUS Database Server administrators to specify whether COMMIT processing is allowed and how the database server should check for currency errors. (The SU Profile is discussed Chapter 2, *Operating the FOCUS Database Server*)

## Controlling COMMIT Processing on the FOCUS Database Server

By default, the FOCUS Database Server accepts MODIFY procedures from source users that include the COMMIT and ROLLBACK subcommands. Maintain procedures require SET COMMIT = ON.

## Controlling Currency Checking on the FOCUS Database Server

By default, the FOCUS database Server determines the FOCURRENT value by checking the segment updated and all parent segments in the path. If a segment in a path was changed between the time a source user retrieves the record and the time they issue a COMMIT, FOCURRENT displays a non-zero value and the COMMIT operation fails.

This default can be changed through the SET PATHCHECK command so that only the updated segment will be checked.

## Syntax

### Setting PATHCHECK

Syntax for the SET PATHCHECK command, which is placed in the SU profile, is:

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

where:

ON

The default, checks parent segments as well as the updated segment to determine the value of FOCURRENT. This will occur for all source users.

OFF

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

---

## CHAPTER 4

# Host Language Interface (HLI)

### Topics:

- Gaining Access to FOCUS Database Servers
- The File Communication Block (FCB)
- HLI Error Codes
- Closing Centrally Controlled Databases
- Writing Transactions from the Buffer
- Using HLI Control commands

This chapter describes how to use the FOCUS Host Language Interface (HLI) to read and modify centrally controlled databases. Through HLI, programs written in COBOL, FORTRAN, PL/1, or Assembler can be used to read and modify the data in FOCUS data sources. This chapter describes the use of HLI with Simultaneous Usage to enable multiple users to read and modify FOCUS data sources concurrently. Each user can retrieve, add, delete, or change data independently.

All programs address the same copy of HLI, which is executed by the FOCUS Database Server. When programs on source machines issue FOCUS calls containing HLI commands, the FOCUS subroutine transmits them to the copy of HLI on the FOCUS Database Server. That in turn executes these commands. It is not necessary to change programs designed to run outside of SU, although certain parameters must be inserted into the File Communication Block as described in *The File Communication Block (FCB)* on page 4-2.

This chapter assumes that readers are familiar with HLI.

## Gaining Access to FOCUS Database Servers

To access FOCUS Database Servers you must first allocate a special communication data set before entering SU. This communication data set was created when SU was installed at your site. You allocate a communication data set for each FOCUS Database Server to be used by providing the following parameters:

- A ddname of your choice.
- The communication data set name of the file allocated to ddname FOCSU by the FOCUS Database Server (see *Chapter 2, Operating the FOCUS Database Server*).
- A disposition of SHR.

To employ two FOCUS Database Servers (SYNCA and SYNCB), you might have the first FOCUS Database Server allocate a communication data set called SYS1.SU01.DATA to ddname FOCSU and the second allocate a communication data set called SYS1.SU02.DATA to FOCSU. You would then allocate these data sets on your TSO ID with the following ALLOC statements:

```
ALLOC F(SYNCA) DA('SYS1.SU01.DATA') SHR
ALLOC F(SYNCB) DA('SYS2.SU02.DATA') SHR
```

To enable other users to also allocate this file, you must allocate it in SHR mode. Note that you refer to an individual FOCUS Database Server (for example, in the SINKID parameter of the FCB and in the ? SU query) by the ddname to which you allocated its communication data set. In the above example, you would refer to the first FOCUS Database Server with ddname SYNCA and to the second with ddname SYNCB.

Do not allocate the centrally controlled databases. Allocate all other files as you would if running HLI programs to read and modify local databases. (**Note:** HLI users wishing to use other databases locally must remember to allocate them with DISP=OLD).

## The File Communication Block (FCB)

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

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

The FCB has a fixed layout of 200 bytes. The user program supplies some FCB parameters, such as a filename for the database, and HLI fills in the others, such as the status return code.

When running a program under SU, place the value SU in FCB word 6, followed by two trailing blanks. Place whatever ddname you assigned the communications dataset associated with the FOCUS Database Server in FCB words 9 and 10. These are the only changes needed to make your programs run under SU.

## FCB Area Layout

This table describes the layout and contents of the File Communication Block:

<b>FCB Words</b>	<b>Item</b>	<b>Meaning</b>	<b>Bytes Used</b>
1-2	FN	Filename of FOCUS data source	8
3-5		Reserved	12
6	SU	SU followed by 2 trailing blanks	4
7-8	PROCNAME	Displayed on HLIPRINT	8
9-10	SINKID	Ddname of Communication file	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 (integer) of NEWSEG	4
24	STATUS	Status return code (integer)	4
25	ERRORNUM	Error code (integer)	4
26-32		Reserved	28
33	SHOLENGTH	Length of one returned record	4
34	LENGTH	Length of work area returned	4
35-50		Reserved	64

### *File Communication Block Layout*

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

## Using Both Local and Centrally Controlled Databases

A single program may use a combination of centrally controlled databases and local FOCUS data sources. When a program is using centrally controlled databases, word 6 in the FCB must contain the characters SU followed by two blanks, and the communication dataset ddname must appear in words 9 and 10. Under HLI, each open file must have its own FCB.

## HLI 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 zero (0) code means that the command executed correctly. A status code value of 1 indicates that the command executed correctly, but that no more data of the type requested was available. All higher status codes indicate error conditions such as incorrect fieldnames or missing parameters. A list of status codes appears in Appendix C, *HLI Status Codes Returned in FCB Word 24*.

IF you are modifying the database using the INP, CHA or 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. 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.)

**Note:** Status code 802, detail error code 5, indicates that the FOCUS Database Server terminated abnormally.

## Closing Centrally Controlled Databases

Before your program finishes running it should issue the CLO command for any FCBs that are open. If it does not, HLI closes the databases for you (even if your program ends abnormally).

If another user closes a file that you are also using, the FOCUS Database Server saves both their changes and any changes you made up to that point.

## Writing Transactions from the Buffer

When users make changes to the database using HLI (using the CHA, INP or DEL commands), HLI stores their changes in a buffer or shadow page until one of the following events occurs.

- The buffer becomes full.
- A program finishes running.
- A program closes a FOCUS file (the CLO command).
- A program issues a SAV command\*

When any of these events occurs, *all pending changes for all programs* are written to the database (even if someone else's program issued the CLO or SAV command).

## Using HLI Control commands

HLI users can add control commands to their programs to control the operation of the FOCUS Database Server.

### Syntax

### Using HLI Control Commands

HLI control commands are of the form:

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

where:

*fc*

Is a File Communication Block. The name of the control command to be executed must be placed in words 1 and 2 of the FCB, instead of the filename parameter (see *FCB Area Layout* on page 4-3).

The HLI control commands are shown in this table:

Command	Command Function
STOPSINK	Stops the FOCUS Database Server normally, executing queued requests and closing all open files.
NX	Stops the FOCUS Database Server cold. Because this command can leave files vulnerable to damage, it should be used only in an emergency.
ECHO	Activates the trace facility that writes information to the HLIPRINT file. This command must be placed in word 1 of the FCB. The FOCUS Database Server only writes trace information for HLI programs that place the word ECHO or STAT in word 18 of their FCBs

To use the ECHO command, put one of the following values into word 2 of the FCB:

Word 2 values	Effect

0	The FOCUS Database Server will only write trace information for programs that place the words ECHO or STAT in word 18 of their FCBs.
1	The FOCUS Database Server will write trace information for all commands. Equivalent to starting the Server with the ECHO parameter in word 18 of the FCB.
2	Turns off the trace facility
3	The FOCUS Database Server will write extended trace information for all commands. Equivalent to starting the Server with the STAT parameter in word 18 of the FCB.

Before using the HLI control commands, you must set the FCB words as follows:

Words	Contents
6	Must contain SU followed by two trailing blanks (the blanks are required)
9 and 10	Must contain the ddname of the communication data set allocated by the source machine. The FOCUS Database Server must have this data set allocated to ddname FOCSU
19 and 20	Must contain the password used as the first parameter to the program HLISNK when the FOCUS Database Server was started. If the FOCUS Database Server was started without a password, you cannot use control commands.

---

## APPENDIX A

# Improving SU Performance - Technical Notes

### Topics:

- Storing Central Databases on Disk
- Additional Suggestions

This appendix is for the systems support staff and contains suggestions to improve the performance of the Simultaneous Usage facility. Note that the larger the application and the more users on it, the more important it is to design the central database for maximum efficiency.

## Storing Central Databases on Disk

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

- Store your central databases on disk packs that have little activity besides SU.
- Store them on volumes not used for paging or swapping and do not store them on a system volume.
- Store each database on a separate volume if possible. If this is not possible, try to distribute the databases across as many volumes as possible.
- Do not drive the volumes higher than their normal I/O rates. For stated device I/O rates, see the appropriate IBM publication.

## Additional Suggestions

Here are three additional suggestions for improving SU performance:

- Give each FOCUS Database Server a performance priority higher than, or at least equivalent to a TSO user.
- Place different databases on different FOCUS Database Servers.
- Placing FOCUS Database Servers in separate domains or making them non-swappable improves performance at some sites.

---

## APPENDIX B

# Error Messages

### Topics:

- Accessing Errors Files
- Displaying Messages Online

If you need the text or explanation for any error message, either display it online in your FOCUS session or look in one of the standard FOCUS ERRORS files.

## Accessing Errors Files

For OS/390 and MVS FOCUS there are eight members of the ERRORS PDS:

FOT004 ERRORS  
FOG004 ERRORS  
FOM004 ERRORS  
FOS004 ERRORS  
FOA004 ERRORS  
FSQLXLT ERRORS  
FOCSTY ERRORS  
FOB004 ERRORS

## Displaying Messages Online

To display messages online, issue the following query at the FOCUS command level:

? *nnn*

where:

*nnn*

Is the number of the error message.

The complete error message number (FOC*nnn*) is then displayed along with an accompanying explanation of the message (if one exists). For example, issuing the following command:

? 500

displays the following message:

(FOC500) SU. COMMAND NOT RECOGNIZED:

An invalid HLI command was issued. For example: NXT rather than the NEX command.

---

## APPENDIX C

# HLI Status Codes Returned in FCB Word 24

### Topics:

- HLI Status Codes

This appendix lists HLI status codes and their meanings.

## HLI Status Codes

Code	Meaning
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 (e.g., NXT rather than the NEX command).
761	Too few arguments have been provided in the HU 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 cannot be found by the sink machine.
765	The requested Master File 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 (e.g., 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.
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.
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.

<b>Code</b>	<b>Meaning</b>
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.
777	The external security system (ACF2, RACF, or TOP SECRET, for example) has determined that you do not have access rights to a file controlled by a sink machine.
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 NXI) 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 NXK or NXI) may not be deleted.
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 F03, 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).

<b>Code</b>	<b>Meaning</b>
789	Attempt to change the instance has not been performed because it has already been changed by another user. (Analogous to FOCURRENT=3 in MODIFYISU.)
790	SU cannot be used with more than 1500 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.
793	You cannot update, delete or include records for a file that has been specified as read only for use with Multi-Threaded HLI/SU Reporting.
794	You cannot declare a file to be read via Multi-Threaded HLI/SU Reporting if it has been opened by another FCB not operating Multi-Threaded HLI/SU Reporting. You cannot open a file that will not be read via Multi-Threaded HLI/SU Reporting if it has already been opened by another FCB operating under Multi-Threaded HLI/SU Reporting.
798	The master file description read by the central database 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 if the sink machine is still active and, if not, restart it.
804	The central database 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

## A

access rules, 2-17  
accessing a FOCUS Database Server, 4-2  
allocating the communication data set, 3-2

## B

BINS parameter, 2-5

## C

CACHE parameter, 2-5  
calculating core requirements, 2-14  
centrally controlled data sources, 1-2, 3-2  
change/verify protocol, 1-3  
    managing simultaneous updates, 1-3  
CHECK, 3-18  
    vs. COMMIT and ROLLBACK, 3-18  
checkpoint facility, 3-12  
CLO command, 4-4  
closing centrally controlled databases, 4-4  
COMMIT, 3-12, 3-13  
    and ROLLBACK vs. CHECK, 3-18  
communication data set, 3-2  
conflicting update transactions  
    testing with FOCURRENT, 3-8  
core requirements, 2-14

## D

data sources, 3-2  
    centrally controlled, 3-2

## E

ECHO command, 4-5  
    values, 4-5  
error messages, 2-18  
    files containing, B-2  
    files displaying online, B-2  
ERRORS.DATA allocation, 2-2

## F

FCB, 4-2  
    changes required for SU, 4-2  
    for multi-threaded HLI/SU reporting, 3-3  
    layout, 4-3  
    required for reporting/updating, 3-3  
File Communication Block (FCB), 4-2  
file update conflicts, 3-8  
FocCurrent, 3-7  
FOCLIB.LOAD allocation, 2-2  
FOCSU communications data set, 2-2  
FOCURRENT, 3-7, 3-8  
    evaluating update conflicts, 3-8  
    testing for rejected transactions, 3-9  
    use in validating turnaround fields, 3-9  
    use with COMMIT and ROLLBACK, 3-18  
    values assigned, 3-8  
FOCUS Database Server, 2-2  
    accessing the server, 4-2  
    calculating core requirements, 2-14  
    communications data set (FOCSU), 2-2  
    ERRORS.DATA allocation, 2-2  
    FOCLIB.LOAD allocation, 2-2  
    HLIERROR allocation, 2-3  
    HLIPRINT allocation, 2-3  
    logging activity, 2-7  
    managing users, 2-17  
    MASTER.DATA allocation, 2-3

FOCUS Database Server (*continued*)  
operations, 2-2  
sample JCL, 2-4  
stopping the server, 2-6  
SUSI security interface, 2-16  
using COMMIT and ROLLBACK, 3-14

FOCUSSU, 2-2  
allocating and formatting, 2-2

## H

HLI, 4-2  
buffer, 4-4  
error codes, 4-4  
status codes, C-2

HLI commands, 4-4, 4-5  
CLO, 4-4  
ECHO, 4-5  
NX, 4-5  
SAV, 4-4  
STOPSINK, 4-5

HLIERROR allocation, 2-3

HLIKX (stopping the server), 2-6

HLIPRINT, 2-3  
allocation, 2-3  
creating and using, 2-11  
file contents, 2-8  
file contents (extended format), 2-10  
logging database server activity, 2-7  
reports, 2-12

HLISNK server job, 2-2

Host Language Interface, 4-2

## I

improving performance, 3-17

## J

JCL for FOCUS Database Server, 2-4

## L

logging database server activity, 2-7

losing data, 3-12

## M

managing SU, 3-5  
operations, 3-12  
transactions, 3-13  
users, 2-17

MASTER.DATA allocation, 2-3

messages, B-2  
displaying online, B-2  
files containing, B-2

multi-threaded SU reporting facility, 3-3

multi-user reporting facility, 1-5

## N

NX command, 4-5

## P

PATHCHECK parameter, 2-6  
setting, 3-19

performance, 3-17  
improving, 3-17, A-2

## R

reporting against shared data sources, 3-3

restricting usage, 2-18

ROLLBACK, 3-12, 3-13

## S

SAV command, 4-4

- SET parameters, 2-5
    - BINS, 2-5
    - CACHE, 2-5
    - PATHCHECK, 2-6
    - SUSI, 2-6
    - SUTABSIZE, 2-6
  - simultaneous updates, 1-3
  - Simultaneous Usage, 1-2
    - definition, 1-2
    - source machines, 1-2
  - sink machine (FOCUS Database Server), 2-2
  - source machines, 1-2
    - communicating with multiple servers, 1-3
    - number permitted, 1-3
  - status codes for HLI, C-2
  - stopping the FOCUS Database Server, 2-6
  - STOPSINK command, 4-5
  - SU, 1-2
    - FOCUS version/release restrictions, 1-3
    - improving performance, A-2
    - multi-user reporting facility, 1-5
  - SU profile, 2-5
    - SET BINS, 2-5
    - SET CACHE, 2-5
    - SET PATHCHECK, 2-6
    - SET SUSI, 2-6
    - SET SUTABSIZE, 2-6
  - SU reporting facility, 3-3
  - SU restrictions
    - managing users, 2-18
  - SUSI, 2-16
    - internal operation, 2-16
    - SU security interface, 2-16
    - using the interface, 2-16
  - SUSI parameter, 2-6
  - SUTABSIZE parameter, 2-6
    - managing users, 2-18
- T**
- testing for rejected transactions, 3-9
- U**
- update transactions, 3-12
    - avoiding loss due to system failures, 3-12
- V**
- validating CRTFORM turnaround fields, 3-9
- W**
- writing transactions from the HLI buffer, 4-4

---

# 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, x**3207**.

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

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

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

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

Comments:

---

# Reader Comments