TIBCO WebFOCUS®

Simultaneous Usage Reference Manual for z/OS

Release 8207.27.0
March 2021
DN1000014.0321

Contents

Chapter 1

Introduction

This chapter introduces the concept of Simultaneous Usage (SU), explains what Simultaneous Usage is, and describes how it works.

In this chapter:

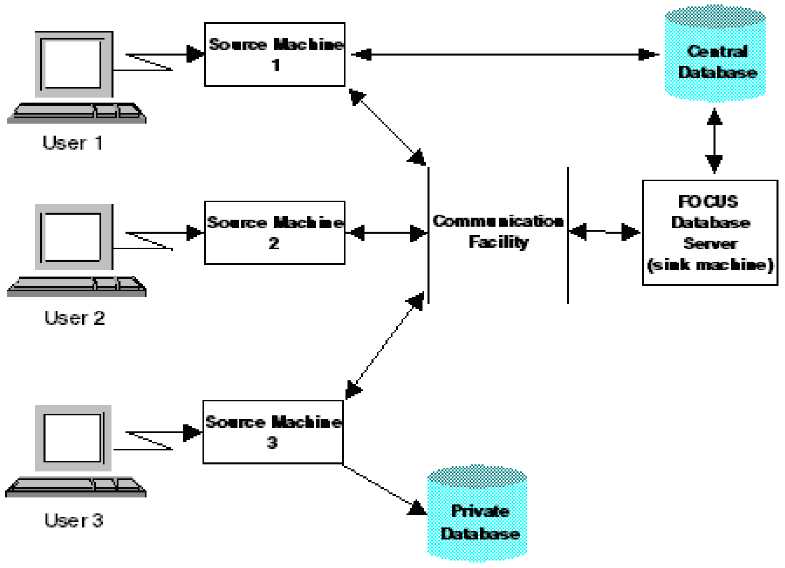
What is Simultaneous Usage?

Multiple users can read and change a FOCUS or XFOCUS 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* or *sink machine*. TSO IDs, MSO sessions, and batch jobs running FOCUS, as well as programs using HLI, that send database retrieval and update requests to the FOCUS Database Server are all called *source machines* or *clients*. 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.

In the remainder of this manual, the terms *FOCUS Database Server*, *server*, and *sink machine* will be used interchangeably as will the terms *source machine* and *client*.

The following representation of SU shows three source machines (a TSO user, a batch job, and an HLI application) 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 z/OS 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).



SU is not needed when users are simply reading a database (for example, issuing TABLE requests), but it 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 in *The Multi-Threaded SU Reporting Facility* on page 8.

Note the following:

- ❏ SU enables up to 512 source machines (users and batch jobs) to change a data sources on a single server concurrently. You can change this number during FOCUS/SU installation.
- ❏ One source machine can communicate simultaneously with as many FOCUS Database Servers as can fit in its USE list.
- ❏ Source machines can communicate with any lower-release FOCUS Database Server.

How SU Processes Transactions Without COMMIT and ROLLBACK

When you submit a transaction that changes a segment 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 how the change/verify protocol interacts with your requests for changes to database records 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 or Maintain 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_ID PAY_DATE
MATCH EMP_ID
  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 procedure, 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 is rejected and SU signals a conflict.

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. Users can always repeat the transaction after they are made aware of the changes that were made by other users. 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. For details, see *Using the Multi-Threaded SU Reporting Facility* on page 35.

Chapter 2

Operating the FOCUS Database Server

This chapter describes how to operate the FOCUS Database Server. The chapter is addressed to system and database administrators and others who want to operate a server.

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

In this chapter:

Starting a FOCUS Database Server

The IBI Subsystem must be running before you can start SU. If the Subsystem was not started, you will get a U0003 system error.

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 data set is allocated to ddname ERRORS.

- The communication data set is allocated to ddname FOCSU. 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 server 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 and a disposition of SHR:

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

  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, copy the Master File named FOCUSSU from FOCCTL.DATA to MASTER.DATA and allocate MASTER.DATA to ddname MASTER, start a FOCUS session, and issue the CREATE FILE command.

Syntax: How to Create the FOCUSSU Dummy Database

```
CREATE FILE FOCUSSU
```

- The FOCUS Database Server writes diagnostic information to ddname HLIERROR in the event of 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 small work file to ddname FOCSORT.
- Allocate the data sources that the FOCUS Database Server will process. If you will use the Multi-Threaded SU Reporting Facility, you must allocate these data sources with DISP=SHR.

- Allocate the Partitioned Data Sets containing the Master Files for the FOCUS and XFOCUS data sources, concatenated to ddname MASTER. One of these PDSs must contain member FOCUSSU. This Master File can be found in the data set FOCCTL.DATA. You should copy this member to MASTER.DATA.
- Allocate the partitioned data set containing the HLI profile to ddname FOCEXEC.

Example: Sample JCL for Running the FOCUS Database Server

```
//*      Job card goes here
//HLISNK   EXEC PGM=HLISNK[,PARM='parameters']
//STEPLIB  DD   DSN=prefix.FOCLIB.LOAD,DISP=SHR
//FOCLIB   DD   DSN=prefix.FOCLIB.LOAD,DISP=SHR
//ERRORS   DD   DSN=prefix.ERRORS.DATA,DISP=SHR
//*
//*      FOCSU IS THE COMMUNICATION FILE
//*
//FOCSU    DD   DSN=prefsu.FOCSU.DATA,DISP=SHR
//*
//*      FOCUSSU IS A DUMMY FOCUS DATABASE CREATED DURING SU INSTALLATION
//*
//FOCUSSU  DD   DSN=prefsu.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)
//*
//*      HLIPRINT IS THE LOG FILE (HERE, THE SYSTEM PRINTER)
//*
//HLIPRINT 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 FOCUSSU
//*      IS IN MASTER.DATA. ONLY DATA SETS WITH IDENTICAL LRECL
//*      AND RECFM ATTRIBUTES SHOULD BE CONCATENATED
//*
//MASTER   DD  DSN=hlq.MASTER.DATA,DISP=SHR
//         DD  DSN=prefix.MASTER.DATA,DISP=SHR
//*
//*      FOCEXEC IS FOR THE SU PROFILE
//*
//FOCEXEC  DD  DISP=SHR,DSN=hlq.FOCEXEC.DATA
//         DD  DISP=SHR,DSN=prefix.FOCEXEC.DATA
//*
//*     CENTRALLY CONTROLLED FOCUS DATABASES ALLOCATED WITH DISP=SHR SO
//*     MULTI-THREADED SU REPORTING FACILITY CAN BE USED.
//*
//EMPLOYEE DD  DSN=prefix.EMPLOYEE.FOCUS,DISP=SHR
//EDUCFILE DD  DSN=prefix.EDUCFILE.FOCUS,DISP=SHR
//JOBFILE  DD  DSN=prefix.JOBFILE.FOCUS,DISP=SHR
```

where:

`parameters`

Are needed if you use the ECHO or STAT facilities or if you plan to use HLI control commands.

`prefix`

Is the high-level qualifier for your production FOCUS data sets.

`prefsu`

Is the high-level qualifier for data sets specific to the FOCUS Database Server.

`hlq`

Is the high-level qualifier for the private data sets of a specific user.

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. For discussions of these reports, see *Logging FOCUS Database Server Activity in HLIPRINT* on page 16.

Using the SU Profile

The Simultaneous Usage Profile (SU Profile) enables users to set parameters for use with SU. This profile is member HLIPROF in a PDS allocated to ddname FOCEXEC in the FOCUS Database Server startup JCL. The DCB attributes 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 XFOCUSBINS = nn
SET CACHE = {nn|NONE}
SET SUTABSIZE = {nnn|32}
SET SUSI = {ON|OFF}
SET COMMIT = {ON|OFF}
SET PATHCHECK = {ON|OFF}
SET SUWEDGE = {n|ddname} [,ddname ...]
```

where:

`SET BINS`

`SET XFOCUSBINS`

A maximum of 1023 bins (which are shared by all users) is allowed. Set BINS or XFOCUSBINS to the maximum for optimal performance.

If BINS/XFOCUSBINS 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 and SET XFOCUSBINS commands for their own processing. Including these commands in the FOCUS Database Server profile affects only simultaneous usage processing.

`SET CACHE`

Cache memory buffers FOCUS or XFOCUS database pages between disk and BINS, reducing disk I/O. Issuing SET CACHE in the profile keeps the entire database in memory, improving performance. The default is no cache memory. See your FOCUS documentation for using cache memory with FOCUS files.

Note that 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`

The FOCUS Database Server has the ability to check authorizations through an external security package on z/OS. This facility is activated through the SET SUSI=ON command in the profile.

The default value is OFF. The FOCUS SU Security interface is discussed in *Protecting FOCUS Databases: the FOCUS/SU Security Interface (SUSI)* on page 27.

`SET SUTABSIZ`

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 COMMIT`

Specifies whether the FOCUS Database Server accepts source users whose MODIFY or Maintain procedures include the COMMIT and ROLLBACK subcommands. For more information, see *SU and the FOCUS Language* on page 31.

`SET PATHCHECK`

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).

`SET SUWEDGE`

Keeps Master Files in memory on FOCUS Database Server even though no users are accessing the files. This enhances performance by eliminating repeated parsing of Master files on FOCUS Database Servers that have a large number of users running multiple applications using different files on the server.

You can specify that a number of files be wedged open, that specific named files be wedged open, or a combination of both. The maximum number of files that will be wedged open is 128, regardless of how many you specify. For complete information, see *Improving Performance* on page 61.

Commands read from the profile and any errors are written to ddname HLIERROR. If HLIERROR is not allocated, the commands and errors are written to HLIPRINT.

The following error message applies specifically to the SU profile:

`(FOC725) THE HLI PROFILE CONTAINS AN ERROR`

Only certain SET commands may be issued in the HLI profile. Either a non-SET line was found, or an invalid SET command was issued. Check the member HLIPROF of the FOCEXEC dataset 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 allocated to ddname FOCSU in the FOCUS Database Server startup JCL.

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 allocated to ddname FOCSU in the FOCUS Database Server startup JCL.

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. For information, see *SU and the Host Language Interface (HLI)* on page 55.

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 (allocated to 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.

You can generate an extended form of the HLIPRINT file by using 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 (Simple Format)

The following is a sample of the HLIPRINT file generated 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
```

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).

`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 sample HLIPRINT file lists a file allocated to ddname FOCUSSU. 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 command failed to locate a segment instance (ON NOMATCH condition), or if a NEXT command reached the end of a segment chain (ON NONEXT condition) this value is set to 1. Otherwise, it is the FOCURRENT field or the HLI status code for the action. For a list of the HLI status codes, see *HLI Status Codes Returned in FCB Word 24* on page 71.

`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 for a FOCUS database is a 4096-byte physical record. A page for an XFOCUS database is a 16K-byte physical record.

`USERID`

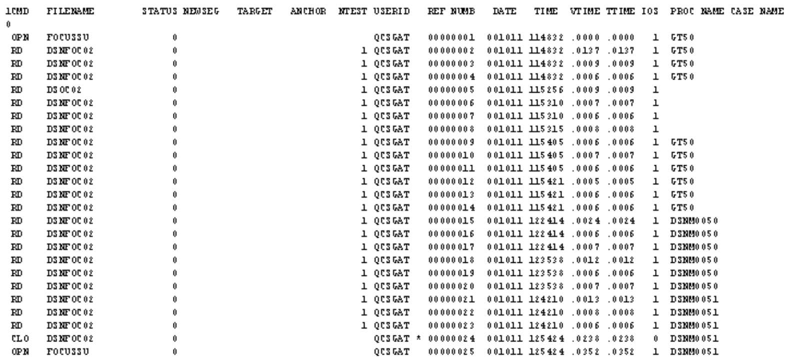
The TSO user ID or job ID that issued the action.

`REF NUMB`

The action number in the HLIPRINT file. REF NUMB is useful with the FOCUS ? FILE command. This query displays a chart showing when each segment in a FOCUS database was last changed (this command is described in the *Developing Applications* 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 (Simple Format)* on page 17, with the additions described following the sample. If you display the file on a system editor, note that the first column is reserved for print control characters.



```
1CMD  FILENAME          STATUS NEWSEG  TARGET  ANCHOR NTEST USERID   REF NUMB  DATE   TIME   VTIME TTIME IOS  PROC NAME CASE NAME
0
 OPN  FOCUSSU              0                              QCSGAT   00000001 001011 114832 .0000 .0000  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000002 001011 114832 .0137 .0137  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000003 001011 114832 .0009 .0009  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000004 001011 114832 .0006 .0006  1  GT50
 RD   DSOC02               0                            1 QCSGAT   00000005 001011 115256 .0009 .0009  1
 RD   DSNFOC02             0                            1 QCSGAT   00000006 001011 115310 .0007 .0007  1
 RD   DSNFOC02             0                            1 QCSGAT   00000007 001011 115310 .0006 .0006  1
 RD   DSNFOC02             0                            1 QCSGAT   00000008 001011 115315 .0008 .0008  1
 RD   DSNFOC02             0                            1 QCSGAT   00000009 001011 115405 .0006 .0006  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000010 001011 115405 .0007 .0007  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000011 001011 115405 .0006 .0006  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000012 001011 115421 .0005 .0005  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000013 001011 115421 .0006 .0006  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000014 001011 115421 .0006 .0006  1  GT50
 RD   DSNFOC02             0                            1 QCSGAT   00000015 001011 122414 .0024 .0024  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000016 001011 122414 .0006 .0006  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000017 001011 122414 .0007 .0007  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000018 001011 123538 .0012 .0012  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000019 001011 123538 .0006 .0006  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000020 001011 123538 .0007 .0007  1  DSNM0050
 RD   DSNFOC02             0                            1 QCSGAT   00000021 001011 124210 .0013 .0013  1  DSNM0051
 RD   DSNFOC02             0                            1 QCSGAT   00000022 001011 124210 .0008 .0008  1  DSNM0051
 RD   DSNFOC02             0                            1 QCSGAT   00000023 001011 124210 .0006 .0006  1  DSNM0051
 CLO  DSNFOC02             0                              QCSGAT * 00000024 001011 125424 .0238 .0238  0  DSNM0051
 OPN  FOCUSSU              0                              QCSGAT   00000025 001011 125424 .0352 .0352  1  DSNM0051
```

A Master File is supplied for the extended HLIPRINT file (member HLIPRINT in MASTER.DATA on the distribution tape). This enables you to write FOCUS reports against the file. For details on producing these reports, see *Producing HLIPRINT Reports* on page 23.

`DATE`

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

`TIME`

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

`VTIME`

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 server 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 or XFOCUS 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 issuing 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), it is 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 following command, the CASE NAME column will display the value MARK until the next COMPUTE command places another value in the SUPRINTNAME field:

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

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.

Creating and Using an HLIPRINT File

To record activity in the HLIPRINT file, follow these instructions when starting the FOCUS Database Server:

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 startup JCL for the server. 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.

   The following is a sample DD card 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 card 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 card 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, disk, or a SYSOUT queue.

3. Use the following EXEC card 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.

`password`

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 log facility has an associated Master File called HLIPRINT. This enables you to produce summary reports of your HLIPRINT log 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 log file. If you named your HLIPRINT data set MYTRACE.HLIPRINT.DATA, you would issue the following command from within FOCUS:

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

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

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

The following sample request determines how much of the server resources were used by each user ID:

```
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 user ID.

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
```

The output is:

```
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
```

This report shows that 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 Memory 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 memory 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 users.

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

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 USAGE attribute in the Master Files. Integer and floating-point fields are four bytes long, packed fields can be eight or 16 bytes long, and double-precision fields are eight bytes. Include in this sum the lengths of cross-reference fields (that is, fields in segment types 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 client 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 at least one client 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 the 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=Sl, 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 (2x48) + (7x72) + 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 memory requirements for a FOCUS Database Server. Assume, for example, that a maximum of 20 data entry clerks can be on the FOCUS Database Server at one time and that 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 Memory Requirements When HLI Is Not in Use

If no client 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. Assuming, for example, that a maximum of 20 data entry clerks can be on a FOCUS Database Server at the same time, the user overhead is:

```
4K x 20 = 80K
```

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

```
620K + 250K + 4K + 80K = 954K
```

Example: Calculating Memory Requirements When HLI Is in Use

If at least one client is using HLI, you add the total field lengths of the two databases (450 + 750), which gives 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. Assuming, for example, that a maximum of 20 data entry clerks can be on the Database Server at a time, the user overhead is:

```
7620 x 20 = 152,040 bytes, or about 152K
```

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

```
620K + 250K + 4K + 152K = 1026K
```

Improving Performance on page 61 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®, CA-ACF2®, or CA-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 *z/OS 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 in the server startup JCL.

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

```
//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 FOCUS Database Server 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 flies are also checked for read access.

For TABLE requests, a read-access authorization check is done each time FOCUS reads page 1 of a database. This includes 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 z/OS security packages using the RACINIT and RACHECK variants of the RACROUTE macro. These functions trigger security routine execution through the System Authorization Facility (SAF), rather than by native RACF. Since security packages, such as RACF, CA-ACF2, and CA-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 data set). In this case, the 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 server rejects the request from the source user.

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 a 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 server 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 active list for the external security system 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 entries the table. You can control the number of entries in the user table through the command SET SUTABSIZE.

Syntax: How to Set the Maximum Number of Entries in the User Authorization Table

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

where:

nnn

Is a number from 2 to 256. The default value is 32.

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

Note that the use of this command does not affect the number of users who can simultaneously access the server. 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.

Authorization 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 (CA-ACF2, RACF, or CA-TOP SECRET) determined that the user has 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, status code 777 appears in HLIPRINT. HLI/SU users get status code 777 in word 24 of the File Communications Block when they issue an OPN command for a file for which they do not have access.

Rejected security requests are logged in the FOCUS Database Server job output stream. In addition, rejected SU security requests are 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 allocated to ddname HLIERROR (or HLIPRINT if HLIERROR is not allocated), but for security purposes, they are not displayed at the user terminal.

If the Interface 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 server.

If HLISECUR is invoked but the exit was not activated with the SET SUSI=ON command, no indication is given and the FOCUS Database Server provides no access checking.

Chapter 3

SU and the FOCUS Language

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

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

- The SCAN and FSCAN facilities are read-only.
- If you combine databases with the COMBINE command, all databases must be controlled by the same FOCUS Database Server.
- Users cannot use the REBUILD utility or the RESTRICT command, or issue the CREATE command for a database controlled by the FOCUS Database Server.

In this chapter:

Using Centrally Controlled Databases

When using centrally controlled data sources, you communicate through the FOCUS Database Server that controls them. So, before issuing requests to SU, you must allocate a communication data set for each FOCUS Database Server you will access. 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 FOCUS Database Server.

- A disposition of SHR.

Assume, for example, that you will use two FOCUS Database Servers. One server has communication 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 could 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 you allocated their communication data sets. In this example, the first server is SYNCA and the second is SYNCB.

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

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

Syntax: How to Access Centrally Controlled Databases

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

```
USE
fileid1 ON server
fileid2 ON server
.
.
END
```

where:

fileid1, fileid2

Are ddnames for FOCUS databases you will access on the sink machine.

server

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

In the USE command, you must specify all of the databases you will use under SU, but you can also specify locally controlled databases.

Example: Accessing Centrally Controlled Databases

In this example, the EMPLOYEE and EDUCFILE databases are controlled by the server whose communication data set is allocated to ddname SYNCA, and JOBFILE is controlled by the server whose communication data set is allocated to ddname SYNCB:

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

Example: Accessing Both Centrally Controlled and Local Databases

The following USE command accesses the EMPLOYEE database controlled by the server whose communication data set is allocated to ddname SYNCA, the EDUCFILE database controlled by the server whose communication data set is allocated to ddname SYNCB, and the PRODUCT database allocated in the client address space:

```
USE
EMPLOYEE ON SYNCA
EDUCFILE ON SYNCB
PRODUCT
END
```

Notice that several FOCUS Database Servers can be active at the same time.

Syntax: How to Specify Additional USE Options for Centrally Controlled Databases

ON may be used with AS in the USE command if the keyword READ is included. The syntax is:

```
USE action
fileid [READ|NEW] [AS mastername]
```

or

```
fileid AS mastername ON server READ
```

or

```
fileid LOCAL
```

or

```
fileid ON server
.
.
.
```

or

```
fileid AS mastername ON server READ
```

or

```
fileid LOCAL
```

or

```
fileid ON server
.
.
.
END
```

where:

action

Is one of the following:

ADD

Appends one or more new file IDs to the present directory. If you issue the USE command without the ADD parameter, the list of data sources you specify replaces the existing USE directory.

CLEAR

Erases the USE directory. The keyword END is not required with this option. Any other options specified will be ignored.

REPLACE

Replaces an existing file ID in the USE directory. This option enables you to change the file specification for the file ID and the options following the file ID.

fileid

Is the ddname of a FOCUS or XFOCUS database.

READ

Restricts data sources to read-only access.

NEW

Indicates that the data source has yet to be created.

AS *mastername*

Specifies the name of the Master File to be associated with the file ID.

Note: When using the AS phrase for files on the FOCUS Database Server with ON, the READ parameter is required. Write access to these files is prevented, as they are being accessed with Master Files other than those with which they were created. Only READ access is allowed in this case.

`ON` *`server`*

Specifies the ddname of the sink machine communication data set.

Note: When using the AS phrase for files on the FOCUS Database Server with ON, the READ parameter is required. Write access to these files is prevented, as they are being accessed with Master Files other than those with which they were created. Only READ access is allowed in this case.

`LOCAL`

This option requires a previous directory entry for the file ID with the ON *server* option. Use the LOCAL keyword to use the Multi-Threaded SU Reporting Facility.

The following options after the file ID are valid together:

```
READ and AS
NEW and AS
AS and ON and READ
```

Any other combination of options after the file ID is not valid.

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 this facility, you must allocate the centrally controlled data sources in SHR mode in the client address space. For example:

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

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

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 communication data set for the FOCUS Database Server and the database are catalogued under high level qualifier 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
   ```

   or

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

3. Set up the FCB as follows:

   a. Set FCB word 6 (SU) to SULO.

   b. Set FCB word 9 (SINKID) to the ddname of the server communication data set.

For example, assume FOCSU is the communication data set and the following is the 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 FCB_SINKID.
```

In the USE command for using the Multi-Threaded SU Reporting Facility, each database must be declared twice, once with the ON option and once with the LOCAL option. For example:

```
USE
CAR ON FOCSU
CAR LOCAL
EMPLOYEE ON FOCSU2
EMPLOYEE LOCAL
.
.
.
END
```

In the example, the two FOCUS databases, CAR and EMPLOYEE, 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 which 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 changes made by another user, 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 changes submitted by other users, keep the following in mind:

- To find out if someone else is using your data source through 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

The FOCUS Database Server can issue error messages. In addition, you can query the server to generate a list of users on the server.

Error Messages

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

When you send a request or transaction to a server that is not operating, you receive one of the following error messages:

- ❏ 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 server is shut down normally while you are in the middle of executing a MODIFY or Maintain request, all your transactions are written to the database. If the server stops abnormally, some of your last transactions may be lost (to minimize these losses, see *Protecting FOCUS Database Server Transactions* on page 46).

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

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

Listing Users on a FOCUS Database Server From a Source Machine

You can see who is logged on to a FOCUS Database Server and the ddnames of the databases they are using.

Syntax: How to List Active Users on a FOCUS Database Server

Issue 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: Listing Active Users On a FOCUS Database Server

Assume user WIBMLH allocated the communication data set for a FOCUS Database Server to ddname SYNCA. To query active users on the server, the user issues the following query command from TSO ID WIBMLH:

```
? SU SYNCA
```

If there are active users on the server, the output will be similar to the following:

```
USERID  FILEID   QUEUE
WIBMLH  EMPLOYEE
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:

```
There Are No Users Connected to Server SYNCA
```

If SYNCA is not operating, one of the error messages described in *Error Messages* on page 38 would display.

Testing the Status of the Server With the ? SU Query

You can use the ? SU query to test the return code variable &RETCODE in FOCEXEC procedures. When ? SU is executed, the query results determine the value stored in &RETCODE. Query results and their corresponding values are:

Query Result	&RETCODE Value
Users are active on the server.	0
No users are active on the server.	8
The server is not running.	16

You can later test the &RETCODE value in a -IF command. For example, this procedure tests whether the server is available before starting a MODIFY or Maintain procedure:

```
? SU ddname
-RUN
-IF &RETCODE EQ 16 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 the value of &RETCODE is 0 or 8, the specified MODIFY or Maintain procedure will be executed.

Using MODIFY or Maintain With SU

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

- The FOCURRENT field, which is an automatically generated temporary field available in MODIFY and MAINTAIN that indicates if your transaction was rejected because of a conflict.
- Using the FOCURRENT field to test for conflicts.
- The FOCURRENT field 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 Field

When you submit a MODIFY or Maintain transaction in SU, FOCUS stores a code in a field called FOCURRENT. This field indicates whether or not there is a conflict with another transaction, as determined by the change/verify protocol (explained in *Introduction* on page 5). If the field value is 0, there is no conflict and the transaction is accepted. If the value is not 0, there is a conflict. In this case, the transaction is rejected and an error message is issued.

FOCUS treats a MODIFY or Maintain transaction that has been rejected because of a conflict as if it failed a validation test (the VALIDATE command). You can log these transactions using the LOG INVALID command described in the *Maintaining Databases* manual.

You can also design your MODIFY or Maintain requests to test FOCURRENT and branch according to its value. For example, a MODIFY or Maintain request can submit a transaction, test FOCURRENT, and, if FOCURRENT is not zero, resubmit the transaction. This technique is explained in *Testing for Rejected Transactions* on page 42.

The values assigned to FOCURRENT are:

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

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

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

In summary, the FOCURRENT field is not 0 if:

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

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

An example of this particular situation is shown in *Validating CRTFORM Turnaround Fields (MODIFY Only)* on page 43.

Testing for Rejected Transactions

By testing the FOCURRENT field, MODIFY or Maintain requests can process transactions even after they have been rejected because of conflicts. You design these requests using Case Logic. Case Logic is discussed in the *Maintaining Databases* manual.

Example: Resubmitting a Rejected Transaction

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

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

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

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

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

If the instance with the employee ID is still there, it prompts 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 (MODIFY Only)

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 the field has been updated by someone else.

Turnaround fields are fields entered on the CRTFORM with a prefix (T.fieldname) used to 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 more information on turnaround fields, see the *Maintaining Databases* manual.)

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

Therefore, your MODIFY request should test the FOCURRENT field after you retrieve 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 of the instance, FOCURRENT contains the value 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 employee salaries, allowing each employee a maximum salary of $50,000 yearly:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID:   <EMP_ID"
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH CRTFORM LINE 2
      "ENTER SALARY: <T.CURR_SAL"
   ON MATCH VALIDATE
      SALTEST = IF CURR_SAL LE 50000 THEN 1 ELSE 0;
   ON INVALID TYPE
      "INVALID SALARY: PLEASE CORRECT VALUE"
   ON MATCH UPDATE CURR_SAL
   ON MATCH IF FOCURRENT NE 0 GOTO ERROR;
CASE ERROR
TYPE
   "FOCURRENT TEST FAILED"
   "ANOTHER USER HAS CHANGED THIS SEGMENT INSTANCE"
GOTO TOP
ENDCASE
DATA VIA FIDEL
END
```

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

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

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

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

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

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

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

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

When you press Enter, these values are entered into the database, canceling the update made by the other user (which may be correct).

To solve the problem, have the request test the FOCURRENT field when it retrieves copies of instances from the database. The previous MODIFY request could be written in the following manner (additions are in bold):

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID:   <EMP_ID"
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH IF FOCURRENT NE 0 GOTO ERROR;
   ON MATCH CRTFORM LINE 2
      "ENTER SALARY: <T.CURR_SAL"
   ON MATCH VALIDATE
      SALTEST = IF CURR_SAL LE 50000 THEN 1 ELSE 0;
ON INVALID TYPE
      "INVALID SALARY: PLEASE CORRECT VALUE"
ON MATCH UPDATE CURR_SAL
ON MATCH IF FOCURRENT NE 0 GOTO ERROR;
CASE ERROR
TYPE
   "FOCURRENT TEST FAILED"
   "ANOTHER USER HAS CHANGED THIS SEGMENT INSTANCE"
GOTO TOP
ENDCASE
DATA VIA FIDEL
END
```

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

If you execute the previous request and enter an invalid salary, FOCUS clears the turnaround fields and notifies you of the failed validation test as before. But when you press Enter again, FOCUS tests the FOCURRENT field. If another user has changed the value of SALARY after you retrieved it, the reMATCH on EMP_ID returns a FOCURRENT value of 1. If FOCURRENT is not 0, CASE ERROR informs you that your transaction has failed the FOCURRENT test and requests that you re-enter the EMP_ID value. Only when you enter the EMP_ID value again does FOCUS retrieve a fresh copy of the instance, displaying the updated salary value:

```
ENTER EMPLOYEE ID:              071382660
ENTER SALARY:                   13000.00
```

You can now decide to leave this updated value or change it.

Protecting FOCUS Database Server Transactions

The following facilities are available to protect transactions in case a system failure brings down the FOCUS Database Server:

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

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

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

It is sometimes a good idea to log all transactions in a sequential file so that they may be re-entered if any are lost. To do this, add LOG or TYPE commands to the MODIFY or Maintain request. LOG commands are satisfactory for simple requests, while TYPE commands are better for complex Case Logic requests. For more information, see the *Maintaining Databases* manual.

You can also protect the integrity of the centrally controlled database itself with the Absolute File Integrity feature described in your FOCUS documentation. Absolute File Integrity is not needed for Maintain.

The Checkpoint Facility (MODIFY Only)

When FOCUS accepts transactions, it does not write the transactions to the database immediately. Rather, it collects them in a buffer. When the buffer is full, FOCUS writes them all to the database at the same time. This reduces the number of input/output operations that FOCUS must perform. However, if your z/OS system fails and brings down the server, the transactions collected in the buffer may be lost. You can instruct FOCUS to write more frequently to the database by using the Checkpoint facility, described in the *Maintaining Databases* manual.

The Checkpoint facility is activated with the MODIFY CHECK command that specifies how many transactions to accumulate in the buffer before writing them to the database. The following request writes transactions to the database in groups of 10:

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

Managing MODIFY and Maintain Transactions: COMMIT and ROLLBACK

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

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

The MODIFY CHECK command only enables you to control the number of transactions that must occur before changes are written to the database. When using CHECK, you cannot change the Checkpoint setting once the request begins execution. Similarly, changes cannot be cancelled. For information on the CHECK command, see *The Checkpoint Facility (MODIFY Only)* on page 46.

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

Maintain uses FOCURRENT to report the result of the latest COMMIT action. Maintain does not support the Checkpoint facility. Therefore, the only way to control a unit of work in Maintain is with the COMMIT and ROLLBACK subcommands.

Absolute File Integrity is required in order to use COMMIT and ROLLBACK in MODIFY. 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. A logical transaction is a group of database changes in the MODIFY or Maintain environment that you want to treat as one. For example, you can handle multiple records displayed on a CRTFORM and then processed using the REPEAT command as a single transaction. A logical transaction is terminated by either COMMIT or ROLLBACK. COMMIT and ROLLBACK also can be used for single record processing.

When COMMIT ends a logical transaction, it writes all changes to the database. Once changes have been committed, they cannot be rolled back. COMMIT can be coded as a global subcommand or as part of MATCH or NEXT logic. The possible MATCH and NEXT commands are:

```
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 changes to the database. The ROLLBACK subcommand cancels changes made since the last COMMIT. ROLLBACK cannot cancel changes once a COMMIT has been issued for them.

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

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

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

Using COMMIT and ROLLBACK With the FOCUS Database Server

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

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

For example, User A and User B execute a MODIFY or Maintain procedure against the EMPLOYEE database on a 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. 2. Updates current salary to $50,000. 3. Issues a COMMIT command.	1. Matches on employee ID 123. 2. Deletes employee 123. 3. Issues a COMMIT command.

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

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

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

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

The BANK file in this application is a FOCUS database 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    ,$
```

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

1. The CRTFORM requests the account from which to take funds, the account to which funds will be transferred, and the amount to be transferred.
2. CASE SUBTRACT subtracts the amount from the first account.
3. If the account is not in the database, any pending changes that may have already been made are rolled back.
4. The new balance is checked before an update takes place. If the balance is less than zero, the user goes to the top case. No UPDATE or COMMIT occurs at this point.
5. CASE ADD adds the amount to the second account.
6. If the account is not in the database, any pending changes that may have already been made are rolled back. This is particularly important in CASE ADD, because funds may have already been subtracted (the UPDATE in CASE SUBTRACT).

7. The COMMIT is executed if both updates are successful.
8. A test for a zero value in FOCURRENT is placed after the COMMIT command. If the COMMIT fails (FOCURRENT has a non zero value) the transaction will be automatically rolled back.

Note:

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

Performance Considerations

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

Comparing COMMIT and ROLLBACK to the Checkpoint Facility

When using CHECK processing, the FOCUS Database server buffers all changes to BINS before saving your changes to the database. The changes are saved from BINS to disk when a checkpoint is issued using the CHECK command in MODIFY. For example, CHECK 5 tells the MODIFY procedure to issue a checkpoint after every five changes to the database.

The CHECK facility has several limitations. There is little control over when checkpoints are invoked. When one user on a FOCUS Database Server issues a checkpoint, the server saves pending changes from all other users. Also, changes cannot be undone.

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 server. No one else sees your changes until you issue a COMMIT. The changes can be cancelled by the ROLLBACK command.

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

Using FOCURRENT

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

When you use COMMIT and ROLLBACK, you should test FOCURRENT after each COMMIT. If the COMMIT fails, FOCURRENT is set to a non-zero value. MODIFY and Maintain only check FOCURRENT after a COMMIT, not after every transaction. If the value of FOCURRENT is not zero, MODIFY or Maintain rolls back the entire logical transaction.

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

Viewing Changes Made by Other Users While Using COMMIT and ROLLBACK

Changes by other users are only visible after a COMMIT has been issued and processed successfully. If you issue a MATCH or NEXT command on a record more than once and you have not issued a COMMIT, the MODIFY or Maintain does not go to the server again. This means that you will not see changes made by other users until your MODIFY or Maintain executes a COMMIT command, although you will see your own changes. Once a COMMIT is performed, the record will come from the database if you issue a MATCH or NEXT on it.

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

SU Profile Commands for COMMIT and ROLLBACK

Two commands are available for use in the SU profile that enable the FOCUS Database Server administrator to control whether COMMIT processing is allowed and how the server checks for currency errors. (The SU Profile is discussed in *Using the SU Profile* on page 13.)

Syntax: How to Control COMMIT Processing on the FOCUS Database Server

By default, the FOCUS Database Server accepts source users whose MODIFY or Maintain procedures include the COMMIT and ROLLBACK subcommands. The following command in the SU Profile controls whether the server accepts these procedures.

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

where:

`ON`

Causes the server to accept MODIFY and Maintain procedures that issue COMMIT and ROLLBACK commands or that set COMMIT ON. ON is the default value. Maintain procedures require SET COMMIT = ON.

`OFF`

Causes the server to reject procedures that either issue COMMIT and ROLLBACK commands or set COMMIT ON.

Syntax: How to Control Currency Checking on the FOCUS Database Server

The server, by default, determines the value of FOCURRENT by checking the segment that was updated and all of the parent segments in the path. If any segment in a path was changed between the time a source user retrieved the record and the time the user issues a COMMIT, FOCURRENT has a non-zero value and the COMMIT fails.

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

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

where:

`ON`

Checks parent segments as well as the updated segment to determine the value of FOCURRENT. Occurs for all source users. ON is the default value.

`OFF`

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

Chapter 4

SU and the Host Language Interface (HLI)

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

All programs address the same copy of HLI, which is executed by the FOCUS Database Server. When a program on a source machine issues a FOCUS call containing an HLI command, the FOCUS subroutine transmits the call to the copy of HLI on the server, which then executes the command. You do not need to change programs designed to run in local mode. However, you must insert certain parameters into the File Communication Block. For more information, see *The File Communication Block (FCB)* on page 56.

This chapter assumes that you are already familiar with HLI.

In this chapter:

Gaining Access to Centrally Controlled Databases

To access databases controlled by a FOCUS Database Server you must allocate a communication data set before entering SU. For information about allocating the communication data set on the server and on the client, see *Starting a FOCUS Database Server* on page 9 and *Using Centrally Controlled Databases* on page 31.

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.

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 in the call, followed by 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 characters SU in FCB word 6, followed by two trailing blanks. Place whatever ddname you assigned the communication data set 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

FCB Words	Item	Meaning	Bytes Used
1-2	FN	Filename of a FOCUS database(or HLI control command)	8
3-5	FT	Reserved	8
6	SU	SU (SU followed by two blanks)	4
7-8	PROCNAME	Displayed on HLIPRINT	8
9-10	SINKID	Ddname of communication data set	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

FCB Words	Item	Meaning	Bytes Used
21-22	NEWSEG	Name of highest new segment retrieved	8
23	SEGNUM	Segment number of NEWSEG (integer)	4
24	STATUS	Status return code (integer)	4
25	ERRORNUM	Detail error code (integer)	4
26-32	-	Reserved	28
33	SHOLENGTH	Length of one record returned	4
34	LENGTH	Length of work area returned	4
35-50	-	Reserved	64

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

Using Both Local and Centrally Controlled Databases

A single program can use a combination of centrally controlled databases and local FOCUS databases. Whenever the program is using the centrally controlled databases, the FCB must have the characters SU   in word 6 and the communication data set ddname in words 9 and 10. Under HLI, each open file must have its own FCB.

Error Codes

After every HLI call is executed, a status code is placed in the File Communication Block (FCB) word 24, shared by the host program and HLI. A status code of 0 means that the command executed correctly. A status code of 1 means that the command executed correctly, but no more data of the type requested was available. All higher numbered status codes indicate error conditions, such as incorrect field names or missing parameters. A list of status codes is provided in *HLI Status Codes Returned in FCB Word 24* on page 71.

If you are modifying the database using the INP, CHA, and DEL commands, and another user is trying to modify the same segment instance at the same time, your transaction may be rejected because the transaction submitted by other user was accepted first (this process is explained in *Introduction* on page 5). 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 the FOCUS Database Server terminated abnormally.

Closing Centrally Controlled Databases

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

If another user closes a file you are also using, the server saves both the changes submitted by that user and your changes made up to that point.

Writing Transactions From the Buffer

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

- The buffer becomes full.
- A program finishes execution.
- A program closes down a FOCUS file by issuing 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 the CLO or SAV command was issued by a program from another user).

Using HLI Control Commands

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

Syntax: How to Issue HLI Control Commands

HLI control commands have the following form

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

where:

`fcb`

Is the File Communication Block.

Prepare the FCB as follows:

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

  `STOPSINK`

  Stops the FOCUS Database Server normally, executing queued requests and closing all open files.

  `HX`

  Stops the FOCUS Database Server immediately. 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:

  `0`

  Causes the server to only write trace information for programs that place the words ECHO or STAT in word 18 of their FCBs.

1

Causes the server to 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

Causes the server to write extended trace information for all commands. Equivalent to starting the Server with the STAT parameter in word 18 of the FCB.

- Place the value SU   in FCB word 6 (SU with two trailing blanks).
- Place the ddname of the communication data set allocated by the source machine in FCB words 9 and 10. The FOCUS Database Server must have this data set allocated to ddname FOCSU.
- Place the password used as the first parameter to program HLISNK when the FOCUS Database Server was started in FCB words 19 and 20 (see *Operating the FOCUS Database Server* on page 9). If the FOCUS Database Server was started without a password, you cannot use control commands.

Chapter 5

Improving Performance

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

In this chapter:

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.

Workload Balancing Techniques for Improving Performance

The following additional suggestions may also improve 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.
- ❑ Place FOCUS Database Servers in separate domains or make them non-swappable. This improves performance at some sites.

Keeping Master Files Open on the FOCUS Database Server

The SET SUWEDGE command provides a mechanism for keeping Master Files in memory on a FOCUS Database Server even though no users are accessing the files. This enhances performance by eliminating repeated parsing of Master Files on FOCUS Database Servers that have a large number of users running multiple applications using different files on the server.

You can specify that a number of files be wedged open, that specific named files be wedged open, or a combination of both. The maximum number of files that will be wedged open is 256, regardless of how many you specify.

Syntax: How to Wedge Master Files Open on a FOCUS Database Server

Place one or more of the following commands in the server profile (PROFILE HLI):

```
SET SUWEDGE = {n|ddname} [,ddname ...]
```

where:

n

Is a number of Master Files to be wedged open. You can specify any number, but once 256 files are wedged open, additional files will not be wedged. The default is zero.

ddname

Is the ddname of a specific file to be wedged open in addition to the *n* files specified by number (if any).

Note: To set both a number of files to be wedged and one or more specific ddnames to be wedged, you can issue the SET SUWEDGE command multiple times or specify multiple options in one SET SUWEDGE command. Specific ddnames are not counted in the number of files specified (as long as the total does not exceed the maximum number of wedged files allowed). If you specify a number multiple times, the last one specified is the one used.

Reference: Usage Notes for SET SUWEDGE

- The most improvement in performance is gained by wedging open large Master Files and those used most often in MODIFY applications.
- Master Files are not opened by the SUWEDGE facility until a MODIFY or Maintain procedure that references them is executed.
- If a wedged Master File has a static cross reference to another Master File, only the parent Master File is wedged open when it is used in a MODIFY or Maintain application. A cross-referenced file is only wedged open when it is explicitly referenced in a MODIFY or Maintain procedure.

- All Master Files that participate in a COMBINE structure are opened when the COMBINE structure is referenced in a MODIFY application. Similarly, all files participating in a MAINTAIN procedure are opened when the Maintain procedure is executed.
- Using the SUWEDGE facility causes increased use of memory on the FOCUS Database Server.
- The maximum number of Master Files that can be open at one time on a FOCUS Database Server is 256, regardless of wedging.
- Files whose names begin with a number cannot be wedged open by explicitly specifying their names in the SET SUWEDGE command.
- If you specify a number of files to be wedged open and you issue a TABLE request on the FOCUS Database Server, the FOCUSSU file will be wedged open.
- You can see which files have been wedged open by examining the HLIPRINT file. The first OPN command listed for a wedged file will be associated with the user ID *suwedge*.

Example: Wedging Files Open on a FOCUS Database Server

The following commands wedge open any three FOCUS data sources plus the EMPLOYEE and CAR data sources:

```
SET SUWEDGE = 3
SET SUWEDGE = EMPLOYEE
SET SUWEDGE = CAR
```

or

```
SET SUWEDGE = 3
SET SUWEDGE = EMPLOYEE, CAR
```

or

```
SET SUWEDGE = EMPLOYEE, CAR
SET SUWEDGE = 3
```

or

```
SET SUWEDGE = 3, EMPLOYEE, CAR
```

Appendix A

FOCUS Error Messages

This appendix explains how to display the text and explanation for any FOCUS error message and lists FOCUS error messages that apply to SU.

In this appendix:

Accessing Errors Files

All messages are stored in member of the ERRORS PDS. For example:

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

Displaying Messages Online

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

```
? nnn
```

where:

nnn

Is the message number.

The complete error message number (FOCnnn) 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:

`(FOC5OO) SU. COMMAND NOT RECOGNIZED:`

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

Messages and Descriptions

`(FOC500) SU. COMMAND NOT RECOGNIZED:`

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

`(FOC501) SU. COMMAND SYNTAX INCORRECT:`

Too few arguments have been provided in the HLI command.

`(FOC502) SU. FILE NOT OPENED:`

The command cannot be executed because the file has not yet been opened (i.e., no OPN command has been issued).

`(FOC503) SU. MORE MEMORY NEEDED ON THE CENTRAL DATABASE MACHINE:`

The sink machine requires more virtual storage to operate. Restart the sink machine in a larger region.

`(FOC504) SU. MASTER FILE CANNOT BE LOCATED:`

The requested Master File Description cannot be found by the sink machine.

`(FOC505) SU. DESCRIPTION OF DATA NOT FOUND:`

The requested Master File Description cannot be found by the sink machine.

`(FOC506) SU. NO DATA FOUND FOR FILE:`

No data found in the specified file. The file must first be initialized by the CREATE command even if no data is entered.

`(FOC507) SU. ERROR IN FILE DESCRIPTION:`

An error was encountered in the Master File Description on the sink machine.

`(FOC508) SU. INVALID PARAMETER IN THE HLI CALL:`

An invalid parameter was encountered in an HLI call received by the sink machine (for example, NTEST is less than zero).

`(FOC509) SU. FIELDNAME NOT FOUND IN FILE DESCRIPTION:`

The fieldname referenced on a SHO command does not exist in the Master File Description. Check the spelling or the structure of the NAMES array passed into the SHO command.

`(FOC510) SU. FILE IS NOT OPENED. CLOSE IGNORED:`

The file specified in the CLO command is not open. The CLO command has been ignored.

`(FOC511) SU. SEGMENT NAME NOT RECOGNIZED:`

The segment specified as a TARGET or an ANCHOR segment is not found in the Master File Description.

`(FOC512) SU. OPTION NOT SUPPORTED:`

The HLI command passed to the sink is recognized but is not yet supported.

`(FOC513) SU. NO CURRENT POSITION ACTIVE IN FILE:`

No current position has been established from which to execute the command.

`(FOC514) SU. TEST RELATION NOT RECOGNIZED:`

An invalid test relation was passed to the sink machine. Possibly the number of tests was specified incorrectly. Valid test conditions are LT, LE, GT, GE, EQ, NE, CO, and OM.

`(FOC515) SU. IMPROPER USE OF VIRTUAL SEGMENT:`

An attempt was made to change or improperly use a cross reference segment.

`(FOC516) SU. ERROR ON INCLUDE. UNIQUE INSTANCE ALREADY EXISTS:`

An attempt was made to include a second instance of a unique segment for a particular parent instance. The transaction is ignored.

`(FOC518) SU. NO PATH EXISTS FROM THE 'ANCHOR' TO THE 'TARGET' SEGMENT:`

The ANCHOR and TARGET segments specified do not lie on the same path in the file.

`(FOC519) SU. INDEXED FIELD DOES NOT HAVE FIELDTYPE=I:`

An error has occurred in the use of an indexed field, or a field named in an NXD command is not indexed.

`(FOC520) SU. NO VALUE SUPPLIED FOR AN INDEXED LOOKUP:`

On an NXD call, no tests on the target segment were provided.

`(FOC521) SU. NO CURRENT PARENT ESTABLISHED FOR CROSS REFERENCE RETRIEVAL:`

No parent position has been established for the retrieval of a cross reference segment. The key for the linked segment is not active, and so no retrieval can be performed.

`(FOC522) SU. THE PASSWORD DOES NOT PROVIDE FILE ACCESS RIGHTS:`

Check the password provided in the FCB.

`(FOC523) SU. PASSWORD DOES NOT ALLOW THIS ACTIVITY:`

The command issued is not allowed with the current password. Check the password provided in the FCB.

`(FOC524) SU. ATTEMPT TO INCLUDE A DUPLICATE SEGMENT:`

The segment instance identified by the key values is already in the database. The database has not been changed. (Analogous to FOCURRENT=1 in MODIFY/SU.)

`(FOC525) SU. CANNOT DELETE RANDOMLY OBTAINED SEGMENT:`

Segment instances obtained with NXK or NXD may not be deleted.

`(FOC526) SU. CURRENT POSITION HAS BEEN LOST:`

Attempt to change the instance has not been performed, because the instance has been deleted by another user. (Analogous to FOCURRENT=2 in MODIFY/SU.)

`(FOC527) SU. FCB IS STILL OPEN, AND CANNOT BE REUSED:`

An OPN command was issued for an FCB which was already open. Before reusing the FCB, it must be closed.

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

Too many files are open on the central database machine (limit 50) or too many FCBs are open (limit 256).

`(FOC529) SU. CURRENT RECORD HAS BEEN CHANGED BY ANOTHER USER:`

Attempt to change the instance has not been performed because it has already been changed by another user. (Analogous to FOCURRENT=3 in MODIFY/SU.)

`(FOC530) SU. THE NUMBER OF FIELDS IN THE FILE EXCEEDS 1500:`

SU cannot be used with more than 1,500 fields in a FOCUS file, including cross references.

`(FOC531) SU. ALL FILES IN COMBINE DO NOT HAVE SAME CENTRAL MACHINE:`

The FOCUS USE command must designate that all of the files used in the COMBINE have the same sink machine. Either they must all be controlled by the same sink or they must all be local files.

`(FOC535) MORE CORE IS NEEDED TO EXECUTE REQUEST: COMMIT`

The amount of main storage is insufficient for executing the specified command. Define larger storage and re execute the command.

`(FOC538) SU. CENTRAL AND LOCAL USERS HAVE DIFFERENT MASTER DESCRIPTIONS:`

The Master File Description read by the central data base machine is not the same as the one used by the source or local machine.

`(FOC540) SU. ENVIRONMENT PROBLEM IN THE CENTRAL DATABASE MACHINE:`

The central database machine has terminated because of a fatal error.

`(FOC541) SU. CENTRAL DATABASE MACHINE ERROR:`

A communication error has occurred on the central database machine.

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

The central database machine has not been started or the wrong name has been used for the central database machine.

`(FOC543) SU. ERROR COMMUNICATING WITH CENTRAL DATABASE MACHINE:`

A communication error has occurred. Check to see that the sink machine is still active and, if not, restart it.

```
(FOC544) SU. COMMUNICATION NO LONGER AVAILABLE TO
CENTRAL DATABASE MACHINE:
```

The central data base machine is not active. Check to see if it has been terminated, and restart it.

`(FOC545) SU. UNRECOVERABLE CENTRAL DATABASE MACHINE ERROR:`

A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.

`(FOC546) SU. UNRECOVERABLE CENTRAL DATABASE MACHINE ERROR:`

A fatal error has occurred on the sink machine. Check to see if the sink machine is active and, if not, restart it.

`(FOC549) INVALID. A SIMULTANEOUS USER HAS JUST INPUT THIS SEGMENT:`

Another FOCUS user has input a segment with the same keys as you. Re retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.

`(FOC550) INVALID. A SIMULTANEOUS USER HAS JUST CHANGED THIS SEGMENT:`

Another FOCUS user has changed some data in this segment before you. Re retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.

`(FOC551) INVALID. A SIMULTANEOUS USER HAS JUST DELETED THIS SEGMENT:`

Another FOCUS user has just deleted the segment you are trying to update or include. Re retrieve the data and try again, or alter your MODIFY case logic to check the value of FOCURRENT and automatically retry.

`(FOC725) ERROR IN HLI PROFILE:`

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

`(FOC726) OPTION NOT SUPPORTED WITH COMMIT/SU:`

COMMIT/SU does not support the use of an alternate file view.

`(FOC758) NO FILE INTEGRITY AVAILABLE. MODIFY TERMINATED:`

The COMMIT/ROLLBACK feature of MODIFY requires Absolute File Integrity in order to run.

`(FOC759) FATAL ERROR DURING COMMIT. MODIFY TERMINATED:`

A fatal error was encountered while applying the COMMIT to the database.

`(FOC760) FATAL ERROR DURING ROLLBACK. MODIFY TERMINATED:`

A fatal error was encountered while attempting to ROLLBACK the database.

`(FOC761) UNABLE TO INITIALIZE FOCSORT. MODIFY TERMINATED`

In order to run a MODIFY with the COMMIT/ROLLBACK feature, FOCSORT must be used. Check to see if FOCSORT was allocated properly.

`(FOC766) UNABLE TO ESTABLISH SINK POSITION. ROLLBACK FORCED:`

A position from which to match on the sink could not be established. Any further database actions are meaningless. Therefore a ROLLBACK is forced.

`(FOC767) UNABLE TO ESTABLISH NEW COMMIT USER ON THE SINK:`

There is no more room on the sink for another COMMIT user. As a result the user is terminated.

`(FOC769) SU. FUNCTION DISABLED: COMMIT:`

COMMIT functionality on the sink machine has been disabled. Contact your sink machine administrator.

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

Must specify the READ option on a USE statement when using the AS and ON options.

Appendix B HLI Status Codes Returned in FCB Word 24

This appendix lists the HLI status codes that can be found in FCB word 24 after an HLI call.

In this appendix:

Status Codes and Explanations

The following table lists the status codes returned by HLI in FCB word 24.

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 (no more target segment instances within the anchor segment).
760	Command not recognized. An invalid HLI command was issued. For example, NXT rather than the NEX command.
761	Too few arguments have been provided in the HLI command.
762	The command cannot be executed because the file has not yet been opened (no OPN command has been issued).
763	The server requires more virtual storage to operate. Restart the server in a larger region.
764	The requested Master File cannot be found by the server.
765	The requested Master File cannot be found by the server.
766	No data found in the specified file. The file must first be initialized by the CREATE command even if no data is entered.

Code	Meaning
767	An error was encountered in the Master File on the server.
768	An invalid parameter was encountered in an HLI call received by the server (for example, NTEST is less than zero).
769	The fieldname referenced in a SHO command does not exist in the Master File. 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 server is recognized but is not yet supported.
773	No current position has been established from which to execute the command.
774	Test relation not recognized. An invalid test relation was passed to the server. 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 (CA-ACF2, RACF, or CA-TOP SECRET, for example) has determined that you do not have access rights to a file controlled by a server.
778	The ANCHOR and TARGET segments specified do not lie on the same path in the file.
779	An error has occurred in the use of an indexed field, or a field named in an NXD command is not indexed.

Code	Meaning
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 NXD 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 FCB, it must be closed.
788	Too many files are open on the server (limit 4096) or too many FCBs are open (limit 512).
789	Attempt to change the instance has not been performed because it has already been changed by another user. (Analogous to FOCURRENT=3 in MODIFY/SU.)
790	SU cannot be used with more than 1,500 fields in a FOCUS file, including cross references.
791	The FOCUS USE command must designate that all of the files used in the COMBINE have the same server. Either they must all be controlled by the same server 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.

Code	Meaning
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 read by the central data base server is not the same as the one used by the source or local machine.
800	The server has terminated because of a fatal error.
801	A communication error has occurred on the server.
802	The server has not been started or the wrong name has been used for the server.
803	A communication error has occurred. Check to see that the server is still active and, if not, restart it.
804	The server is not active. Check to see if it has been terminated, and restart it.
805	A fatal error has occurred on the server. Check to see if the server is active and, if not, restart it.
806	A fatal error has occurred on the server. Check to see if the server is active and, if not, restart it.

Legal and Third-Party Notices

Index