TIBCO FOCUS®

Overview and Operating Environments

Release 8207.27.0
March 2021
DN1001055.0321

Copyright © 2021. TIBCO Software Inc. All Rights Reserved.

Contents

1. Introduction to FOCUS 15
What Is FOCUS? 15
Who Uses FOCUS? 16
FOCUS Language 16
Terminal Operator Environment 17
FOCUS Concepts 18
Combining Data From Several Data Sources. 21
Features for End Users 22
Report Writer: TABLE. 23
Row-oriented Financial Reports: Financial Modeling Language. 23
Graph Generator: GRAPH. 24
Text Editor: TED or IEDIT. 24
Data Export Interface. 25
Features for Application Developers 25
Data Source Security. 26
Dialogue Manager. 26
Interactive Menus and Windows: Window Painter. 27
Data Source Management: Maintain and MODIFY. 27
Full Screen Data Entry Forms: FIDEL. 29
Data Source Editor: FSCAN. 29
Resource Governor: SmartMode. 29
FOCUS User Aids 30
Using Regular Expressions 31
2. Editing Files With TED 35
Entering TED 35
TED Features 37
Screen Layout. 37
Current Line. 37
Command Line. 38
Moving the Cursor. 38
TYPE Environment. 38
EDIT and Prefix Area Commands. 39

INPUT. 39
PAINT. 40
Creating a File 40
TYPE and EDIT Functions 41
Adding Lines. 42
ADD. 42
==A==, ==I== 43
PF2. 43
CINS. 44
INPUT 44
Moving the Current Line. 44
:n. 44
±n. 44
==/==. 44
CUrline 45
Inserting and Replacing Text. 45
Command Line Commands. 45
REplace. 46
Overlay 46
Input. 47
Deleting and Recovering Deleted Text. 47
==D==. 47
==DD=. 48
DELete. 49
CDel. 49
RECover. 49
Moving Through a File. 50
BAckward. 51
FOrward. 51
PF7, PF8, PF19, PF20. 51
Top, Bottom. 52
DOWN, UP, NEXT. 52
:n. 53

±n. 53
Locating and Changing Text. 53
Command Line Commands. 53
Locate. 53
Change. 54
Copying and Moving Text. 55
COpy. 56
==C==. 56
==CC=, ==""=. 57
==”n=. 58
DUplicat. 58
MOve. 58
==M==. 58
=MM==. 59
Joining and Splitting Text. 59
==J==. 59
Join. 60
=SP==. 60
SPLit. 60
Editing Multiple Files. 60
Command Line Commands. 60
SPH, SPLITH 61
SPV, SPLITV. 62
TEd. 62
Transferring Text Between Files and Temporary Storage. 62
==PP=, PUT, PPUT. 63
==PLn. 64
==PD=, PUTD, PPUTD. 64
==G==, Get. 65
Displaying a Scale and Line Numbers. 66
Command Line Commands. 66
NUm ON. 66
NUm OFF. 67

SCale ON. 67
SCale OFF. 67
Displaying or Repeating the Previous Command. 68
=, PF5. 68
?, PF6. 68
&. 68
Moving the Screen Display. 68
RIght. 68
RIGHTP, PF11. 69
LEft. 69
LEFTP, PF10. 69
Specifying Uppercase and Lowercase Text. 69
Command Line Commands. 69
CAse M. 69
CAse U. 70
UPPercas. 70
LOwercas. 70
Ending a TED Session. 70
Quit, PF3. 71
QQuit. 71
FILe. 71
SAve. 72
FFILE, SSAVE. 72
Accessing the HELP File 72
Editing FOCEXECs 73
Personalizing TED: PROFILE and PFnn 75
Syntax Summary 75
Function Keys. 76
Prefix-Area Commands. 76
Command Line Commands. 77

3. Invoking the ISPF Editor on z/OS 83

Editing Files With IEDIT 83

Installing IEDIT 84
Using IEDIT 84

4. Terminal Operator Environment 87

Illustrating the Terminal Operator Environment 88
Invoking the Terminal Operator Environment 88
Activating a Window 89
Types of Windows 90
Command Window. 90
Output Window. 93
History Window. 93
Help Window: Revising PF Key Settings. 94
Table Window. 95
Error Window. 96
Displaying Fields and Field Formats. 96
Window Commands 97
Commands for Activating a Window. 98
Clearing a Window. 98
Controlling the Output Window. 99
Customizing Your Screen. 100
Displaying the Help Window. 105
Enlarging a Window. 105
Recalling Commands. 105
Routing Window Contents. 106
Scrolling Window Contents. 106

5. UNIX and Linux Guide to Operations 109

How FOCUS Interacts With UNIX 109
Accessing FOCUS. 109
UNIX Directory Permissions. 110
Terminal Support. 110
The FOCUS Operating Environment. 110
Overview of a FOCUS Session 110
The focus Shell Script. 110

Beginning and Ending a FOCUS Session. 110
Profile Processing. 111
Interrupting a FOCUS Session. 111
Defining the FOCUS Operating Environment 112
Defining the Environment Using UNIX Variables. 112
Defining the Environment Using Command-Line Parameters. 112
Defining and Allocating FOCUS Files 113
Allocating FOCUS Files. 113
Dynamically Defining a File Under UNIX and Linux. 115
Assigning a Logical Name With the FILEDEF Command. 116
Displaying Current ddnames Assigned With FILEDEF. 118
Clearing Allocations. 118
Application Files Under UNIX and Linux. 119
Master Files Under UNIX and Linux. 119
Access Files Under UNIX and Linux. 120
Procedures Under UNIX and Linux. 120
FOCUS and XFOCUS Data Sources Under UNIX and Linux. 121
External Indexes for FOCUS Data Sources Under UNIX and Linux. 121
Sequential Data Sources Under UNIX and Linux. 121
FOCUS StyleSheets Under UNIX and Linux. 122
Extract Files Under UNIX and Linux. 122
HOLD Files Under UNIX and Linux. 122
SAVB Files Under UNIX and Linux. 123
SAVE Files Under UNIX and Linux. 123
HOLDMAST Files Under UNIX and Linux. 124
Work Files Under UNIX and Linux. 125
FOCUS in the UNIX and Linux Environments 125
Terminal Support. 126
Remapping Keys. 126
Function Key Support. 127
International 8-Bit Character Support. 127
The Concatenation Symbol. 127
Running FOCUS as a Background Process. 128

Sending Email From a Procedure 128
Configuring, Starting, and Stopping the FOCUS Database Server 133

6. z/OS Guide to Operations 137

Introduction to 64-bit FOCUS 137
Referencing Files 138
Allocating Files. 140
Dynamically Allocating Files. 141
Required Files. 142
Application Files 143
Master Files. 144
Access Files. 145
FOCEXEC Files. 145
PROFILE FOCEXEC. 146
FOCEXECs as Sequential Files. 146
StyleSheet Files. 147
FOCUS Data Sources. 147
Allocating FOCUS Data Sources. 148
Multi-Volume Support. 148
Allocating a Multi-Volume Data Source in TSO and z/OS FOCUS. 152
External Indices for FOCUS Data Sources. 155
MDIs for FOCUS Data Sources. 155
Disposition of FOCUS Data Sources. 155
Database Security: ENCRYPT, DECRYPT and RESTRICT. 156
FOCUS Data Sources and IBM Utility Programs. 157
USERLIB. 158
Window Files 158
Compiled Window Files. 159
Window Transfer Files and Window Documentation Files. 159
Non-FOCUS Data Sources. 160
TTEDIT Files. 160
HOLDSTAT Files. 161
Extract Files 163

HOLD Files. 163
SAVB Files. 164
SAVE Files. 164
Temporary Master Files: HOLDMAST Files. 164
Extract Files That You Allocate: LET, LOG, POST, Dialogue Manager Output Files. 165
Work Files 166
STACK. 166
FOCSORT. 166
FOCSML. 167
FOCPOST. 167
REBUILD. 167
TABLTALK. 168
Enabling Use of the zIIP Specialty Engine 168
What Is a zIIP Specialty Engine?. 169
Steps to FOCUS zIIP Enablement. 169
Obtaining APF Authorization for All Load Libraries. 170
Obtaining AUTHCMD and AUTHPGM Authority for TSO Processing. 172
Activating a zIIP Environment or Projecting zIIP Usage. 172
Establishing a Non-authorized Environment After Enabling the zIIP Feature. 175
How FOCUS Takes Advantage of the zIIP Processor. 177
Evaluating zIIP Usage. 178
Calling FOCUS Under TSO 179
Batch Operation. 179
Direct Entry. 181
FOCUS Facilities Under TSO 183
Using FIDEL. 183
TED Editor. 184
National Language Support. 186
TSO and FOCUS Interaction 187
Issuing TSO Commands From Within FOCUS. 187
Using TSO Commands in FOCUS Applications. 188
FOCUS Command Interrupt Levels. 190
ISPF From FOCUS. 193

ISPF From FOCUS From ISPF. 194
Reviewing Attributes of Allocated Files. 195
DYNAM Command 200
Use of Data Sets. 203
ALLOCATE Subcommand. 203
CONCAT Subcommand. 220
FREE Subcommand. 221
CLOSE Subcommand. 222
COPY Subcommand. 223
COPYDD Subcommand. 226
DELETE Subcommand. 227
RENAME Subcommand. 228
SUBMIT Subcommand. 229
COMPRESS Subcommand. 230
Comparison of TSO Commands, JCL, and DYNAM. 231
Allocating Temporary Files. 232

7. Using FOCUS as a Client to a Reporting Server 235

Client/Server Computing and Middleware 235
Using FOCUS to Access Data on a Server 236
Establishing and Configuring the FOCUS User Environment. 237
Server Configuration File. 238
DNS Names Support. 238
Remote Execution 239
Logging On With REMOTE Commands. 239
Sending Requests to a Remote Server. 241
Using -REMOTE BEGIN and -REMOTE END to Execute Requests Remotely. 242
Viewing a System or Error Message. 245
Terminating the Remote Session: REMOTE FIN. 245
Querying Remote Session Parameter Settings: ? REMOTE. 245
Distributed Execution 246
How Location Transparency Works. 249
Logging On to the Server With Distributed Execution. 250

Joining Data Sources Across Platforms With Distributed Execution. 251
Issuing SQL Commands to the Server With Distributed Execution. 251
Executing Stored Procedures With Distributed Execution. 251
Using SQL Passthru With Distributed Execution. 252

8. Logging FOCUS Usage: FOCLOG 253

Overview of FOCLOG 253
How Logging Is Implemented. 254
The Log Data Set. 255
Sample FOCLOG Configuration Scenarios. 256
Implementing FOCLOG 257
Overview of FOCLOG Implementation. 257
Allocating the FOCLOG Log File. 258
Activating FOCLOG. 260
Validating the FOCLOG Configuration. 262
Running the FOCLOG Reports. 265
Information Captured in the FOCLOG File 265
FOCLOG Reporting 272
Using the Menu-Driven FOCLOG Reporting Interface. 273
Report Layouts. 279
Report Contents. 279
Sort Options. 279
Report Descriptions. 279

9. Configuring FOCUS for National Language Support Services 291

Introduction to FOCUS National Language Support (NLS) Services 291
Overview of Steps for Configuring Code Page Settings 293
Overview of NLS Configuration Files. 294
The FOCUS Default Code Page Configuration. 295
Detailed NLS Configuration Steps 296
Adding New or Alternate Code Pages for FOCUS. 296
Configuring FOCUS To Use a New Code Page. 299
Configuring PC3270 Session Parameters. 299
Verifying the FOCUS Language Configuration. 299

Advanced NLS Configuration Options 300
Configuring Customized FOCUS NLS Monocasing. 300
Configuring Customized FOCUS NLS Sort Sequences. 302
Using the NLS Configuration Files 303
Using the TSGU to Generate New Transcoding Tables 308

10. Unicode Support 313

Unicode Encoding Standards 313
Accessing Unicode Data 315
Selecting, Reformatting, and Manipulating Characters 321
Sort Order Under Unicode 324
Added Unicode Support for Master Files, Data Files, and Application Directory Names 324
Unicode PDF Output 324

Legal and Third-Party Notices 327

Chapter 1

Introduction to FOCUS

The following topics introduce FOCUS, outline its user base, and detail its components and facilities.

In this chapter:

- What Is FOCUS?
- Who Uses FOCUS?
- FOCUS Language
- Terminal Operator Environment
- FOCUS Concepts
- Features for End Users
- Features for Application Developers
- FOCUS User Aids
- Using Regular Expressions

What Is FOCUS?

FOCUS is a complete information control system with comprehensive features for entering, maintaining, retrieving, and analyzing data. It is designed for use both by users with no formal training in data processing and by data processing professionals who need powerful tools for developing complete applications.

The non-procedural FOCUS language is designed to replace traditional programming languages in most application programming situations. The simplicity of the command syntax in the language stems from the fact that it uses simple English phrases that enable most new users to start producing meaningful reports immediately.

Every effort has been made to keep the syntax consistent. As you become more familiar with the products, you will be able to infer how a new feature will work based on your experience using similar features.

Who Uses FOCUS?

FOCUS is designed to serve the needs of both end users and application developers. These two groups have different needs and different levels of data processing experience. End users generally use FOCUS for reporting purposes and to run applications created by others. Application developers create computer systems and design the applications that end users use. Since most FOCUS users quickly advance to designing their own applications, we have a few suggestions for beginners.

If you have never used FOCUS, begin with the FOCUS Report Writing Primer and use the TableTalk and FileTalk tutorials to become familiar with how FOCUS handles basic reporting and data source definition tasks. These facilities present formatted screens from which you select options to create reports and describe data sources.

Depending on your needs and the particular FOCUS options installed with your system, you may also require additional Information Builders publications, including those that describe special adapters. These adapters may access data sources created by other systems or facilities or may allow you to access FOCUS data sources from programs written in other programming languages.

FOCUS Language

The difference between the terms *procedural* and *non-procedural* is worth discussing briefly. The basic distinction is that non-procedural languages allow the person making a request to concentrate on what needs to be done, rather than how to do it. Non-procedural languages free you from the constraints of specifying, in a predetermined way, how to process data. FOCUS takes you a level away from what the computer is doing from moment to moment, allowing you to concentrate on specifying what you wish to accomplish, such as print a report, update a data source, create a graph, or build an entry screen.

Procedural languages, such as COBOL and PL/1, require that you specify how to process the data. For example, to create a simple report showing salaries by department, you would write explicit instructions to:

1. Open the data source.
2. Sort the data source (by DEPARTMENT).
3. Read a record. When there are no more records, go to Summary (below).
4. Extract the values for SALARY and DEPARTMENT.
5. Accumulate field totals.
6. Move the fields to the output positions.
7. Write a record.

8. Go back to read another record.
9. Carry out a summary (write report totals).
10. Close data sources and stop.

The same request in FOCUS might read:

```
TABLE FILE filename
SUM SALARY COLUMN-TOTAL
BY DEPARTMENT
END
```

You can develop highly complex applications in FOCUS, with sophisticated interactive dialogues and processing flows that depend on internal testing of values that you (or another user) supply at run time. These applications comprise non-procedural request elements interspersed with procedural control statements from Dialogue Manager.

The procedures that combine non-procedural request elements and procedural control statements are called FOCEXECs (FOCUS executable procedures), and they can be characterized as quasi-procedural. They still employ the simple request elements, but add procedural control elements to dictate when and under what conditions the request portions will be executed.

In one system, FOCUS provides a convenient means of specifying what you wish to do, together with the procedural controls necessary for building complete applications.

FOCUS consists of several integrated functional environments. TABLE, for example, accommodates commands for requesting tabular reports, while MODIFY works with commands used to add, delete, or change (modify) data.

Each level or environment has a command set that applies specifically to that environment. You will quickly learn to distinguish between environments, but even if you forget where you are, you can have FOCUS display the active environment by pressing *Enter* without typing anything on the command entry line. If you are in FOCUS, but not in a particular command environment, the word "FOCUS:" appears followed by the system prompt symbol.

Terminal Operator Environment

You may run your FOCUS session in the Terminal Operator Environment, an optional environment organized into seven windows. Each window serves a different session function, specifically to:

- Accept FOCUS commands you enter at the keyboard.
- Display your FOCUS session log (a list of commands you entered as well as the FOCUS response to each of them).

- Keep a list of every command you enter for later editing or reuse.
- Redisplay your most recently generated report.
- Display a window of current program function (PF) key settings. You can change a setting by typing over the existing one in the window.
- Display error messages.
- List available fields you can select for use in your request.

The Terminal Operator Environment is described in *Terminal Operator Environment* on page 87.

FOCUS Concepts

Your company probably acquired FOCUS because it maintains a wealth of information that must be organized and made accessible for a variety of uses. The following scenario introduces a few of the concepts and facilities you will use on a daily basis to report from or manage that information.

For example, State University, like most large organizations, maintains information in various places that needs to be coordinated. The following screen contains personal information about students: names, home and campus addresses and phone numbers, and student identification numbers.

```
                        STATE UNIVERSITY
                     PERSONAL INFORMATION
STUDENT I.D. NO:
LAST NAME:
FIRST NAME:                                MIDDLE INITIAL:
HOME STREET ADDRESS:
CITY:                            STATE:           ZIP:
HOME PHONE:
CAMPUS RESIDENCE:                                    ROOM:
CAMPUS PHONE:
```

The areas on the screen where information appears are called entry fields. Each field must have a name that identifies it. For example, LAST_NAME for the last name field or STREET for the street address field. Additionally, each field must be assigned a format to tell the computer whether it is numeric (contains only numeric information and can be used in computations) or alphanumeric (contains a combination of alphabetic and numeric characters and cannot be used in computations; for example, a street address).

Also, the length of each field must be specified so the computer can allocate space for storing the information. A rule of thumb is to specify an exact length (if you know your fields will never exceed that length), or a length slightly longer than the longest entry you anticipate.

Groups of related fields, shown in the previous example, are called segments in FOCUS. Segments have names and can be linked to other related segments. The collected instances of data for one or more related segments constitute a data source. Thus, the collected personal data for all of the students at State University could be gathered in a single segment FOCUS data source.

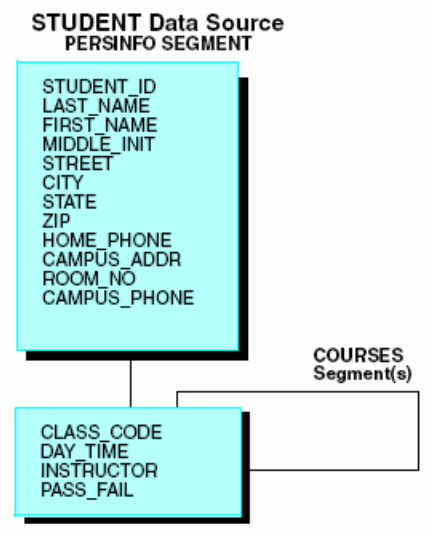
In simple applications, data sources may consist of a single segment, but generally more than one segment is needed. Consider an additional screen that captures information about each course a student selects:

```
                        STATE UNIVERSITY
                     COURSE ENROLLMENT FORM
                        FALL TERM, 2002
STUDENT I.D. NO:
LAST NAME:
FIRST NAME:                                MIDDLE INITIAL: CLASS CODE: DAY/
TIME: INSTRUCTOR:
PASS/FAIL:         Y         N
FOR OFFICE USE:
MIDTERM
FINAL
```

Note that some information on this screen (the student's name and student identification number) also appeared on the Personal Information screen. If we wish to add course information to the personal information already entered for each student, we can do so by adding another segment to our original single segment data source or by creating a second data source.

Since the personal identification fields are already in the first segment, the new segment only needs to contain the fields necessary to describe each course taken (Class Code, Day/Time, Instructor, and Pass or Fail). Let us call the original segment PERSINFO and the new segment COURSES. These segments are related by defining PERSINFO as the parent of COURSES.

Structurally, what we have defined now looks like this:



A single instance of PERSINFO segment data and several instances of COURSES segment data (one per course) will appear in the data source for each student.

The above diagram shows the relationship between fields and segments in FOCUS data sources. You describe this to FOCUS with a Master File, in which you name the data source and each of its segments. In each segment, you name each field and define its format and length. Thus, a Master File defines the complete structure and format of your data. The data itself resides in another file called a data source.

Before you can use FOCUS to write reports, a Master File must exist for each data source you wish to use, regardless of whether the data source is a FOCUS data source or a non-FOCUS data source (created outside of FOCUS).

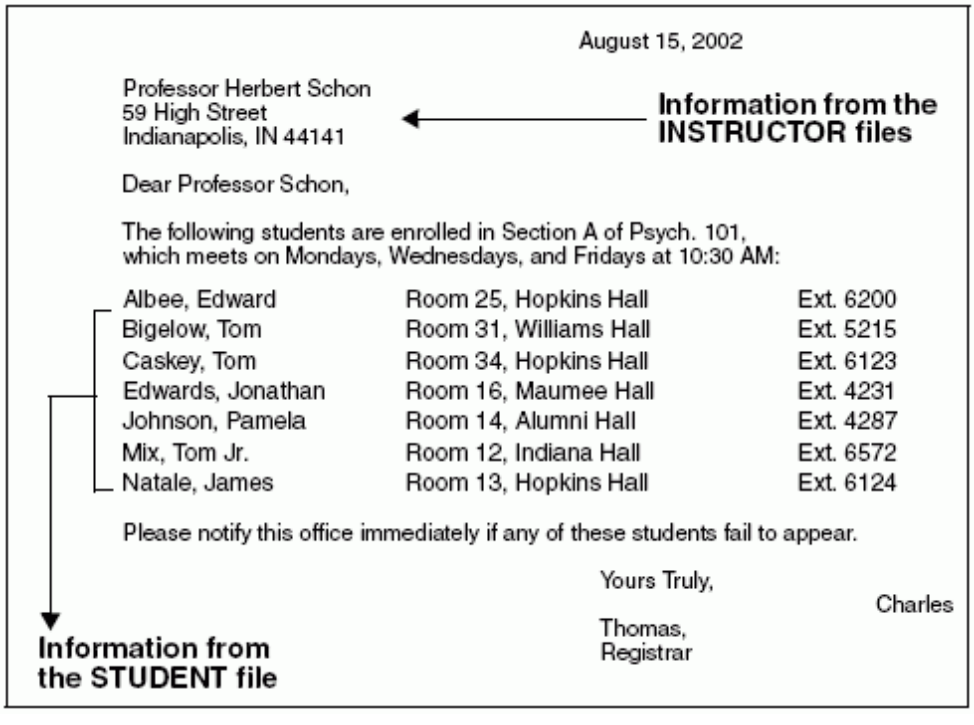
The data (the actual pieces of information described by the entries in the Master File, such as a student's name and Student Identification Number) exist in logical records in the data source. For example, all of the information about a student in the STUDENT data source is in a single logical record. Obviously, there would be many records in the STUDENT data source, one for each student at State University. A student's logical record consists of one instance of data for the PERSINFO segment, describing addresses and identity information, and multiple instances of data for the COURSES segment that describe the individual courses they have selected.

Combining Data From Several Data Sources

FOCUS also includes facilities for joining data sources together, which enables you to include data from several related data sources in a report. For example, suppose State University keeps an INSTRUCTORS data source with information about instructors (names, telephone numbers, and addresses), and the Registrar wishes to send each instructor a letter providing their class schedules and lists of their students' names, addresses, and telephone numbers.

To produce such a letter, you need data from both data sources: the students' names, addresses, and telephone numbers from one data source, and the instructor's address from the other. You can do this using a JOIN operation, in which a common field in both data sources (in this case, INSTRUCTOR) is used to link the two data sources.

The following sample letter, which could be generated by FOCUS, includes data from both of the data sources.



Joined data sources remain physically separate, but FOCUS treats them as a single structure. The JOIN command thus provides a powerful facility for relating data sources. Through it, you have the ability to create new views of data to meet new needs (without prior planning), and you can keep your individual data sources simple and straightforward, making them easy to use and maintain.

See the *Creating Reports* manual for a description of the JOIN command and the specific data sources that can be linked.

Features for End Users

FOCUS provides powerful decision support tools for use by all levels of management. New FOCUS users have started by writing simple report requests against existing data sources. This permits them to be immediately productive while expanding their knowledge of FOCUS.

The following topics are particularly applicable for end users:

- Report Writer, which works with existing FOCUS and non-FOCUS data sources. See the *Creating Reports* manual.
- Financial Modeling Language facility for creating row-oriented financial reports. See the *Creating Reports* manual.
- Full screen text editor (TED or IEDIT) for creating and saving requests, Master Files, and other text files. See *Editing Files With TED* on page 35 and *Invoking the ISPF Editor on z/OS* on page 83.
- Dialogue Manager, a facility for designing and managing applications. See the *Developing Applications* manual.
- Data interface, used to extract specially formatted data (DIF, LOTUS, SYLK, WP) for use with other software products on personal computers.

Each of these facilities is briefly described in the following pages.

Some end users may also be interested in specialized topics described in other publications. These include:

- FOCUS for Mainframe Talk Technology User's Manual, which includes tutorials for TableTalk, FileTalk, ModifyTalk, and PlotTalk.
- *Statistical Analysis User's Manual*.

Report Writer: TABLE

The Report Writer enables you to create reports quickly and easily. It provides facilities for creating highly complex reports, but its strength lies in the simplicity of the request language. You can begin with simple queries and ad hoc requests, and progress to complex reports as you learn about additional facilities.

The data source named in your request can be a FOCUS data source, a collection of data sources related through the JOIN command, or an external file created outside of FOCUS (external files can also be named in a JOIN). In all cases, Master Files must exist for the individual data sources. Master Files for non-FOCUS and FOCUS data sources are described in the *Describing Data* manual.

In TABLE, you have broad capabilities for selecting records, performing calculations, defining special fields, and creating custom report formats. You can report on data from more than one data source at a time and you can specify special handling for records with missing data fields. There are also options for producing a variety of extract files.

Report requests can be typed "live" at your terminal or entered in a named file and then run by executing the file. You can create such files using TED (the FOCUS editor), or IEDIT (a facility for invoking your system editor). These named, executable FOCUS requests are called FOCEXECs (see the *Developing Applications* manual).

Row-oriented Financial Reports: Financial Modeling Language

Financial Modeling Language (FML), formerly known as EMR, is an extension of TABLE specifically designed to handle the special needs associated with creating, calculating, and presenting row-oriented financial data. FML produces financial statements, such as Balance Sheets and Income and Expense Statements.

FML expands the report preparation facilities with facilities for:

- Presenting matrix reports in spreadsheet layouts.
- Performing calculations using the contents of rows and/or columns.
- Carrying column totals forward for use in subsequent reports.
- Incorporating values from external files and special routines.

FML is described in the *Creating Reports* manual.

Graph Generator: GRAPH

The GRAPH command uses the same language and syntax as the TABLE command to produce graphic displays. The Creating Reports manual describes the GRAPH facility. The request statements enable you to perform intermediate calculations and specify grouping and sorting characteristics, and control the format of the graph. A REPLOT command is provided for turning the output of appropriate TABLE requests into corresponding graphs.

You can generate five graph forms with FOCUS (each is defined by using a different combination of request elements):

- Connected point plots
- Histograms
- Bar charts
- Pie charts (on high-resolution devices)
- Scatter diagrams

FOCUS provides a complete set of default graph parameters that establish the lengths and scales of axes for you. All graph elements can be readjusted through SET statements issued before executing the request (or redisplaying it with REPLOT). There are facilities for saving graphs in named files for later production on different plotters or graphics devices.

Text Editor: TED or IEDIT

An optional full screen editor (TED) is available for creating and editing text files for use inside or outside of the FOCUS environment. (You can also invoke your system editor from FOCUS using the IEDIT facility.) In FOCUS, such files can be used as Master Files, or they can store requests for subsequent reuse (FOCEXECs). Outside of FOCUS, TED files can be used for any purpose normally served by system editor files. The TED editor is described in *Editing Files With TED* on page 35. The IEDIT facility is described in *Invoking the ISPF Editor on z/OS* on page 83.

TED is not a word processor; it is a development tool designed to support application building. TED is similar to many system editors in general function, but it has some special features that are particularly useful in FOCUS. Some advantages of using TED, instead of a system editor, include the following:

- It is functionally equivalent in all versions of FOCUS.
- When FOCUS encounters an error while running a stored request, it returns an error message to the terminal. If you then type:

  ```
  TED
  ```

- TED provides direct access to the FIDEL Screen Painter facility, which is used for generating full screen data entry forms.
- TED has split screen facilities, enabling you to display up to four files simultaneously on your screen, and it can move lines from one file to the next.

You can also create and edit comma-delimited or fixed-format data sources with TED. Note, however, that you cannot use it to edit the data in FOCUS data sources. (Use Maintain, MODIFY, or FSCAN to add or edit data in FOCUS data sources.)

Data Export Interface

There are facilities for saving the output of FOCUS requests as formatted data sources for transfer to other machines, for use by other products, or as FOCUS data sources. Specifically, you can:

- Prepare data sources for immediate use by other software packages that may run on a personal computer.
- Automatically create a FOCUS data source and Master File by extracting request output from FOCUS or external files in FOCUS format.

The *Creating Reports* manual describes the facilities for creating extract files.

Features for Application Developers

FOCUS provides a complete application development environment. In addition to the reporting tools, the following features support the development of complete applications:

- Data source security features that offer security at every level, from the data source itself down to specifying protection for specific values in fields. See the *Describing Data* manual.
- Dialogue Manager for building reusable FOCUS requests (FOCEXECs), including facilities for variable substitution, testing and branching, and reading from or writing to external files or the terminal. See the *Developing Applications* manual.
- Facilities for designing menus and windows to select, enter, and display data. See the *Developing Applications* manual.
- Data source management facilities for loading and maintaining data sources. See the *Maintaining Databases* manual.
- Facilities for designing full screen data entry forms, including two screen painters. See the *Maintaining Databases* manual for the Screen Painter and the Winform Painter.

- Online, interactive, full screen data source editing (FSCAN). See the *Maintaining Databases* manual.
- Adapters to other types of data sources, including ADABAS, CA-DATACOM/DB, DB2, Oracle, Teradata, CA-IDMS/DB, IMS/DB, and MODEL 204.
- Host Language Interface for reading FOCUS data sources from programs in other computer languages.
- User-written functions for using subroutines written by other users in other programming languages. See the *Using Functions* manual.

Data Source Security

Access to data in FOCUS data sources and external files can be restricted through FOCUS facilities that permit Database Administrators to select any of the following levels of protection for all or part of each data source:

- No access at all.
- Read-only access.
- Update-only access (add new segments).
- Write-only access.
- Read and write access.

These limits can be varied for individual users, and each user can be given access to entirely different fields or even particular values in fields.

Access rights to data sources are governed through the contents of decision tables associated with the data sources' Master Files.

Dialogue Manager

You can enter and execute FOCUS requests at the terminal, or as text files in an editor for subsequent use whenever you execute the file.

These named, executable requests are called FOCEXECs or stored procedures. They can be created as stand-alone requests or as request procedures that include variable substitution and various interactive prompting sequences. This is the procedural area of FOCUS, and these procedures are the FOCUS equivalent of macros or command lists (CLISTs or EXECs).

The tools for building these procedures are a series of Dialogue Manager control statements or keywords that perform actions such as:

- Sending prompts to the operator. Such prompts typically request values for variable fields.
- Typing messages to the operator.
- Setting and testing values.
- Branching to another area of the procedure or executing nested procedures.
- Reading from, or writing to, an external file or the terminal.

The *Developing Applications* manual describes the Dialogue Manager control facilities. These include facilities for incorporating prompting dialogues in procedures, and including variable fields that are assigned values at run time. These values can be supplied from a variety of sources (typed on the command line, as responses to prompts, through full screen entry forms, in SET statements from external files, or as default values). Therefore, you have a variety of ways to control your processing flow during execution.

Interactive Menus and Windows: Window Painter

You can create a series of menus and windows using Window Painter, and then display those menus and windows on an application screen using the Dialogue Manager -WINDOW statement. When displayed, the menus and windows can collect data by prompting a user to select or enter a value, or press a PF key.

Window Painter enables you to design the menus and windows on the screen, specifying the information offered for selection, prompting, and display. These specifications can include variables that are resolved at execution time. This enables earlier parts of the application to determine the menu selection and information display of later menus and windows.

The Window facility, including Window Painter and the -WINDOW statement, is covered in the *Developing Applications* manual.

Data Source Management: Maintain and MODIFY

To manage data, you may use the graphical Maintain facility or the MODIFY facility.

Maintain is a graphical toolset for building modular data maintenance applications to perform event-driven, set-based processing. Its sophisticated Winform Painter enables you to simply point, click, and type to define forms. You can also include check boxes and buttons to trigger procedures, and display and edit several sets of data simultaneously, moving through each set using automatically provided scroll bars. The Painter generates these forms (Winforms) automatically from your specifications, dramatically reducing the effort required to build an application.

The Maintain language and tools are covered separately in the *Maintaining Databases* manual.

MODIFY invokes the data management environment, which provides complete facilities for the following data source maintenance activities:

- Collecting data.
- Performing validation tests.
- Establishing a position in the data source by matching data against the existing records.
- Performing maintenance actions after establishing a position in the data source, for example, adding records, updating fields, and deleting records.
- Logging data source maintenance activities.

The *Maintaining Databases* manual describes how to incorporate these facilities into MODIFY requests that you can execute to update a FOCUS data source. Such requests can range from simple (containing a few instructions) to extensive (containing multiple full screen entry forms and conditional branching logic, set with values entered at run time). The three following interrelated FOCUS facilities are specifically designed to assist you in preparing data source maintenance procedures:

- MODIFY supplies the data source and record handling facilities, and validation and calculation features.
- Dialogue Manager provides control facilities for creating MODIFY procedures that can include variable fields and prompt for data.
- The MODIFY subcommand CRTFORM passes you to the FOCUS Interactive Data Entry Language (FIDEL) which provides full screen data transfer of field update information.

You can use the JOIN command in the context of MODIFY requests to gain access to data from related data sources, or you can use the COMBINE command in MODIFY requests to update multiple data sources simultaneously.

FOCUS offers two operating environments for data source maintenance:

- A stand-alone system in which a single user modifies a data source at one time.
- A multi-user system in which many users share data sources and modify them simultaneously. This environment is called Simultaneous Usage, and is covered in a separate publication.

Full Screen Data Entry Forms: FIDEL

Data source maintenance and Dialogue Manager procedures require full screen data entry forms for entering information needed to update data sources. The facility for describing screen forms, FIDEL, is discussed in the *Maintaining Databases* manual. You invoke FIDEL by including the keywords -CRTFORM (from Dialogue Manager) or CRTFORM (from MODIFY).

You can describe screen forms with free-form text layout, using spot markers to position the text on the screen. You design windows on the screen, or forms longer than the screen size (up to 1280 lines long), using scrolling features activated with PF keys. Displayed fields can be protected, or left unprotected for updating. FIDEL provides a variety of dynamic attributes for highlighting fields you wish to emphasize, such as blinking fields, background lighting, and colors.

A Screen Painter, entered through TED, generates the FIDEL code and enables you to see your developing screen form and immediately review the effects of your design decisions. This is particularly useful when developing complex screens with many attributes and labels. (The Screen Painter is only available with TED.)

Data Source Editor: FSCAN

The FSCAN facility, described in the *Maintaining Databases* manual, is a data source maintenance utility that enables you to edit FOCUS data sources directly on the screen. FSCAN displays data sources as if they were flat files on a full screen system editor. FSCAN commands allow you to scroll through records, navigate the data source, locate specific field values, and delete records. You can also add new records by typing them in and change field values by typing over them. In addition, FSCAN has these features:

- Prefix areas that enable you to perform an operation on any record on the screen.
- Delete confirmation screens that prevent you from inadvertently deleting records.
- Two modes of operation: one that displays multiple records on the screen, one that displays a single record at a time.

FOCUS also provides a FOCUS data source line editor called SCAN. SCAN is described in the *Maintaining Databases* manual.

FSCAN and SCAN are best suited for making minor changes and corrections to FOCUS data sources. For more extensive maintenance, use the MODIFY or Maintain facilities.

Resource Governor: SmartMode

The SmartMode option is a FOCUS report monitor that predicts resource usage of a report and prevents users from running expensive queries. The System or Database Administrator sets the maximum resource usage permitted for a report.

Using query statistics it has recorded, SmartMode analyzes reports requested with the TABLE, TABLEF, MATCH, GRAPH, and FML (formerly EMR) commands. When SmartMode is governing, it estimates the I/O and CPU resources a request would consume. If the resource usage is greater than the resource usage threshold the administrator has defined, SmartMode cancels the request.

SmartMode provides an interactive administrative facility to control all aspects of its operation. The administrator may establish a different threshold for each shift and mode of execution, and may adjust cost factors to fine-tune the SmartMode protection of the most important resources of the site. Reports and graphs on resource usage patterns show how data sources are used, who uses them, and how resource usage is distributed.

For complete documentation and installation instructions, see the *SmartMode for FOCUS Installation and Operations Manual*.

FOCUS User Aids

A number of easy-to-use facilities are available to FOCUS users across command environment boundaries. These user tools include various query subjects that reveal the current state of the FOCUS environment and facilities for setting the parameters that control the various command environments. Additionally, there is a facility for establishing your own translation table to create substitute command words for some or all of the FOCUS keywords.

FOCLOG is a tool for recording and analyzing FOCUS use for your entire site. It comes packaged with a set of standard analytical reports that allow you to interrogate FOCUS usage—identify usage spikes and redundancies, detect large report requests, analyze time-of-day usage trends, and monitor ad hoc versus scheduled requests for each user. It even allows you to analyze the environmental conditions of the query, such as use of joins, cross references, combines, MSO or SU. In addition, it collects and reports on statistics such as the number of data rows extracted and number of lines on the report output.

FOCREPLAY is a tool that enables you to record your keystrokes when executing an interactive FOCUS application and play them back at some time in the future.

There are also utilities for performing a variety of data source handling tasks: initializing data sources, concatenating them, rebuilding data sources and their indexes, joining data sources together, and transferring data sources from the mainframe to a personal computer.

The following product developed in FOCUS is designed for special use:

- COBOL-to-FOCUS Translator. For those undertaking conversion efforts from COBOL to FOCUS, the Translator converts COBOL FD statements into the equivalent FOCUS Master Files. See the *COBOL FD Translator Users Manual and Installation Guide*.

Using Regular Expressions

Some FOCUS features accept regular expressions as input.

A regular expression is a string containing a tagged expression. A tagged expression is a portion of the string that contains a pattern that will be matched against an actual character string.

This section summarizes how to create a regular expression. More information about regular expressions can be found on the Web.

Reference: Summary of Regular Expressions

Tagged expressions must be enclosed in parentheses. The backslash character (\) is a special character in tagged expressions. If a group ID actually contains a backslash character, you indicate that you want the backslash character to be treated as a normal character by entering two successive backslash characters (\\).

The following can be used to create the tagged expression:

`abc`

Matches *abc* anywhere within the string.

`(abc)`

Matches *abc* anywhere within the string, but the parentheses act as a tag.

`[]`

Defines a character class (or set) that matches any one of the characters in the class. For example, [abc] matches the character a or b or c. The expression [x-y] matches any character from x to y.

`.`

Matches any single character except newline.

`\w`

Matches any word character string (alphanumeric plus "_").

`\W`

Matches any non-word character.

`\s`

Matches any whitespace character.

`\S`

Matches any non-whitespace character.

`\d`

Matches any digit.

`\D`

Matches any non-digit character.

`\t`

Matches a tab character.

`\r`

Matches a return character.

`\f`

Matches a formfeed character.

`\e`

Matches an escape character.

`\b`

Matches a word boundary or a backspace. For example, test\b matches test, but not testing. However, \b matches a backspace character when specified inside a class (that is, [\b]).

`\B`

Requires that there is no word boundary. hello\B matches *hello*, but not hello *there*.

`^`

Matches characters only at the beginning of the string. For example, ^abc matches *abc* at the beginning of the string.

`$`

Matches characters only at the end of the string. For example, abc$ matches *abc* at the end of the string.

`|`

Specifies alternative matching characters. For example, a|b matches either a or b. This metacharacter can also be used with words, for example, abc|def.

`[^abc]`

Matches a character that is not in the set. [^abc]+ will match such strings as *hello*, *test*, and *perl*.

`\`

Is the escape character. For example, * matches the * character. Use the backslash (\) character to escape (remove the special meaning of) characters that have significance in a regular expression.

`(?i)`

Ignores case. For example, (a(?i)b|c) matches *aB*, *c*, and *C*.

You can follow any character, wildcard, or series of characters and/or wildcards with a repetition indicator:

`*`

Matches 0 or more occurrences of the character sequence.

`+`

Matches 1 or 0 occurrences of the character sequence.

`?`

Matches 0 or more occurrences of the character sequence or the shortest match.

`{}`

Is the repetition modifier.

`{n}`

Matches exactly *n* occurrences of the character sequence.

`{n,}`

Matches at least *n* occurrences of the character sequence.

`{n,m}`

Matches at least *n* but not more than *m* occurrences of the character sequence.

Chapter 2

Editing Files With TED

TED is a general-purpose text editor you can use to create or edit files while in FOCUS.

These topics describe the four TED editing environments and their use for creating and modifying Master Files, requests, FOCEXECs, CRTFORMs, and non-FOCUS data sources. For information on creating and maintaining FOCUS data sources, refer to the *Maintaining Databases* manual.

For users familiar with mainframe editors, TED is similar to IBM's ISPF editor that runs under TSO. While that editor can also be used from within FOCUS, TED provides important advantages:

- Moving or copying lines of data from one window to another using the split-screen facility.
- Screen Painter, which automatically generates data entry screens.

Note: TED supports editing of files with LRECL up to 4096.

In this chapter:

- Entering TED
- TED Features
- Creating a File
- TYPE and EDIT Functions
- Accessing the HELP File
- Editing FOCEXECs
- Personalizing TED: PROFILE and PFnn
- Syntax Summary

Entering TED

After entering FOCUS, to enter the TED environment and begin creating or editing files, issue the command

```
TED
```

at the FOCUS command prompt, followed by the name of the sequential file you want to edit or create.

To edit a FOCEXEC, issue the following command on z/OS, where FOCEXEC is the DDNAME for the PDS in which the file resides.

```
TED FOCEXEC(member)
```

On distributed systems, issue the following command.

```
TED filename.fex
```

Or, since FOCEXEC is the default DDNAME for TED on z/OS, and .fex is the default extension for TED on distributed systems, you can issue the following command to edit a FOCEXEC.

```
TED name
```

To edit a file that is not a FOCEXEC on z/OS, issue the following command.

```
TED {ddname[(member)]|'dataset'}
```

where:

ddname

Is the DDNAME of the PDS in which the member resides, the DDNAME allocated to a sequential file, or the fully-qualified data set name of a sequential file enclosed in single quotation marks.

member

Is the member to edit, if the DDNAME is allocated to a PDS.

To edit a file that is not a FOCEXEC on distributed systems, issue the following command.

```
TED name.ext
```

where:

name

Is the name of the file.

ext

Is the file extension, for example .mas for a Master File.

See *z/OS Guide to Operations* on page 137, for more information about referring to DDNAMEs on z/OS.

You can also use TED to enter and edit fields with a text (TX) format. In this case, TED is entered in a MODIFY request (see the *Maintaining Databases* manual).

TED Features

Each of TED's four environments—INPUT, TYPE, EDIT, and PAINT—is discussed briefly here, and more fully in subsequent sections. This section describes the TED screen layout, the concepts of current line and command line, and how to move the cursor within TED.

Screen Layout

When using TED, the following information about the file you are creating or editing is provided on the first line of the screen:

- The data set name, file name, or DDNAME.
- The size (the number of lines in the file).
- The line number of the current line.

The first screen is:

```
IBIMLH.TSOEXAMP.SALES                SIZE=0         LINE=0

* * * TOP OF FILE * * *
* * * END OF FILE * * *



====>
                                                      TED
```

Current Line

The current line is the highlighted line on a screen. The current line is an important concept because most TED functions start with the current line. The line that is current changes during an editing session as you scroll the screen, move up and down, and so forth. Changing the current line is described in *Moving the Current Line* on page 44.

Command Line

At the bottom of the screen there are four equal signs and an arrow. This is the command line. One of the ways you communicate with the editor is by entering TED commands on this line. Commands can be typed in either uppercase or lowercase or a combination of uppercase and lowercase, and may be abbreviated. Also note that no more than one line of text (including commands) can be issued at the command line.

Moving the Cursor

You can use the following cursor control keys on your keyboard to position the cursor on the screen:

↓	Moves the cursor down.
↑	Moves the cursor up.
→	Moves the cursor to the right.
←	Moves the cursor to the left.
Tab	Moves the cursor to the next line in TYPE or to the next Tab stop in EDIT.
BackTab	Moves the cursor to the previous line.
Home	Moves the cursor to the top of the screen.
Return	In TYPE mode, moves the cursor to the command line, In INPUT mode, moves the existing lines up, and stays on the same line, which is now blank, so you can enter additional information.

TYPE Environment

When you enter TED, you are automatically in TYPE (unless you use a TED profile to modify this; see *Personalizing TED: PROFILE and PFnn* on page 75). TED enables you to create lines up to 159 characters long. In TYPE, you can view up to 80 characters at a time.

Furthermore, TYPE provides easy-to-use commands to edit or create files. To use TYPE, simply enter a command at the command line and press the *Enter* key or use one of the many function keys. Commands and function keys are explained in detail in *TYPE and EDIT Functions* on page 41.

Note: To return to TYPE from another TED environment, enter the command TYPE at the command line.

EDIT and Prefix Area Commands

EDIT is similar to TYPE. In both environments, you can create or edit files and use the command line to enter commands. In EDIT, however, you can also use prefix area commands.

The prefix area is the six columns furthest to the left columns on the screen where five equal signs (=====) and a space appear before the lines in the file. Each line in the file has a prefix area associated with it.

You can perform various editing tasks, like deleting lines or moving blocks of text, by entering short commands, called *prefix area commands*, in the prefix area of any line.

To enter EDIT, type

```
EDIT
```

at the command line, and the following screen displays:

```
 FOCEXEC(EXAMPLE)                        SIZE=0     LINE=0

===== * * * TOP OF FILE * * *
===== * * * END OF FILE * * *




====>
                                                EDITING MODE
```

EDIT is fully explained in *TYPE and EDIT Functions* on page 41.

INPUT

Both TYPE and EDIT provide an INPUT mode. INPUT is used for creating files and enables you to type anywhere on the screen without predefining space for a file. To enter INPUT mode, type

```
INPUT
```

at the command line. To return to TYPE or EDIT, press the *Enter* key twice.

PAINT

The FOCUS Screen Painter enables you to create FIDEL screens in a full-screen editing environment, by simply *painting* the screen image. Screen Painter then automatically generates the FIDEL code and places it in your file. For a complete explanation of the PAINT environment and FIDEL, see the *Maintaining Databases* manual.

To access Screen Painter, place a CRTFORM in the file being edited and then enter the following at the command line (or press PF4):

```
PAINT
```

TED scans down the file from the current line until the first CRTFORM statement is found. This statement becomes the current line and invokes Screen Painter.

If you want to call a CRTFORM other than the first one, specify the number of the CRTFORM in the PAINT command. For example, the following accesses the third CRTFORM from the current line:

```
PAINT 3
```

Creating a File

When you enter TED at the FOCUS command line, you are placed in TYPE. Although you may enter data in either TYPE or EDIT, you must first add lines or spaces to accommodate the text you plan to enter. For this reason, TYPE and EDIT are more suited to editing existing files (see *TYPE and EDIT Functions* on page 41).

INPUT, on the other hand, effectively opens the entire screen for entering text. For this reason, INPUT is the best choice for creating new files.

Note: Use INPUT within TYPE or EDIT to enter additional text in existing files by issuing the INPUT command. The additional space starts after the current line of the current file.

To enter INPUT, type *INPUT* at the command line. You can then type text on the screen. For example:

```
 FOCEXEC(EXAMPLE)                      SIZE=0

* * * TOP OF FILE * * *
THE INPUT MODE IS AN EASY WAY
TO ENTER DATA. SIMPLY TYPE THE DATA,
AND PRESS THE TAB OR RETURN KEY TO GO TO THE NEXT LINE.
WHEN YOU HAVE FINISHED, JUST PRESS THE ENTER KEY TWICE,
AND YOU WILL BE BACK IN TYPE OR EDIT MODE.




====> * * * INPUT ZONE * * *
```

Note: When entering text, use the Tab or Return key to move to the next line. When finished, press the *Enter* key twice.

The first time you press the *Enter* key, the screen view scrolls forward so you can type more data on a clear screen. The last line entered becomes the current line, and the cursor is positioned on the line below. When you press the *Enter* key again, TED returns you to your previous environment (EDIT or TYPE) and makes the last line entered the current line:

```
 FOCEXEC(EXAMPLE)                      SIZE=5

===== AND YOU WILL BE BACK IN TYPE OR EDIT MODE.
===== * * * END OF FILE * * *



====>
```

TYPE and EDIT Functions

The following sections describe the various functions within TYPE and EDIT.

In TYPE, you may use function keys or issue commands on the command line.

In EDIT, you can use function keys, command line commands, and prefix area commands. Prefix area commands can be placed anywhere in the prefix area.

Note:

- To cancel pending prefix area operations, use the command RESet.
- You may truncate commands. In the sections that follow, capital letters indicate the shortest acceptable truncation.
- You can also issue the ?F file name and ? nnn commands at the command line. For an explanation of these commands, see the *Developing Applications* manual.

Adding Lines

When creating a file or adding data to an existing one, you must first make space available in the file. The following commands enable you to add lines:

Command Line Commands	Prefix Area Commands	Function Keys
`Add`	`==A==`	`PF2`
`CINS`		`PF2`
`Input`	`==I==`	`Add`

ADD

ADD adds one or more lines into a file after the current line. The syntax is

`Add` *n*

where:

n

Is any number of lines you are adding.

For example, the following screen shows how to add five lines after the current line (the current line, in this case, is the TOP OF FILE line):

```
 FOCEXEC(EXAMPLE)                        SIZE=0     LINE=0

===== * * * TOP OF FILE * * *
===== * * * END OF FILE * * *




====> ADD 5
                                            EDITING MODE
```

After pressing the *Enter* key, five lines are added:

```
 FOCEXEC(EXAMPLE)                        SIZE=5     LINE=0

===== * * * TOP OF FILE * * *
=====
=====
=====
=====
=====
===== * * * END OF FILE * * *



====>
                                              EDITING MODE
```

==A==, ==I==

The prefix area command =An== means to add *n* lines to the file starting with the line in which the command is issued (where *n* can be any number up to 9999). The cursor is positioned to the first new line. =In== is identical to ==An=. If *n* is omitted, the default is line 1.

PF2

To add a single line, position the cursor and press PF2. The new line appears immediately below.

CINS

Inserts a line after the cursor.

INPUT

INPUT enters the INPUT environment.

Moving the Current Line

Most of the commands in this section use the location of the current line as a reference point. For this reason, it is important to know how to move the current line. You can also specify where you want the file to appear on the screen; that is, whether the current line should appear at the top, middle, or bottom of the screen.

The following commands are used to adjust the position of the current line on the screen:

Command Line Commands	Prefix Area Commands
`:n`	`==/==`
`+n`	
`CUrline`	

:n

Enter the colon at the command line, using the following syntax

`:`*n*

where:

n

Is the number of the line you want to make the current line.

±n

Enter a number with a plus sign to move the current line forward or a minus sign to move the current line backward *n* number of lines.

==/==

Enter the slash in the prefix area of the line you want to be the current line. Then, press *Enter*.

CUrline

If you want the current line to be displayed on the top, middle, or bottom of the screen, use the following syntax

```
CUrline n
```

where:

n

Is the number of the line on the screen where the current line will be displayed. To return the current line to the top of the screen omit *n*.

For example, if you issued the command CURLINE 5, the screen would look like this:

```
  FOCEXEC(EXAMPLE)                        SIZE=5     LINE=0




 ===== * * * TOP OF FILE * * *
 ===== THIS SCREEN SHOWS WHAT HAPPENS WHEN YOU USE THE
 ===== CURLINE COMMAND. NOTICE THE FIRST LINE OF THE SCREEN
 ===== IS ON THE FIFTH PHYSICAL LINE OF THE SCREEN.
 =====
 =====
 ===== * * * END OF FILE * * *


 ====>
                                                  EDITING MODE
```

Inserting and Replacing Text

Once you have made space in your file, you can move the cursor to that space and type whatever you want into the file. You can also insert or replace text using the following commands:

Command Line Commands

```
REplace
Overlay
Input
```

REplace

REPLACE completely replaces the text on the current line with a string of character(s) you specify. The syntax is

```
REplace string
```

where:

string

Is the text you want to place on the current line.

Overlay

The OVERLAY command is used to overlay a string of text located on the current line. When you use it, the characters in the new string will be placed on the current line. The new string will only overlay its own length. Unlike the REPLACE command, OVERLAY will not replace the entire text on the current line. The syntax is

```
Overlay string
```

where:

string

Is the string of text that you want to place on the current line without removing existing text.

Note: Only non-blank characters in the string will overlay.

For example:

```
 FOCEXEC(EXAMPLE)                      SIZE=5     LINE=0

===== * * * TOP OF FILE * * *
==/== THIS WILL BE CHANGED TO THE NEXT LINE BUT NOT THIS PART
===== THIS IS AN EXAMPLE OF THE OVERLAY COMMAND
=====
=====
=====
===== * * * END OF FILE * * *




====>  OVERLAY THIS WILL BE CHANGED TO THE LINE ABOVE

                                                    EDITING MODE
```

After pressing the *Enter* key, the following screen appears:

```
 FOCEXEC(EXAMPLE)                        SIZE=5     LINE=1

===== THIS WILL BE CHANGED TO THE LINE ABOVEBUT NOT THIS PART
===== THIS IS AN EXAMPLE OF THE OVERLAY COMMAND
=====
=====
=====
===== * * * END OF FILE * * *



====>
                                                    EDITING MODE
```

Input

The INPUT command allows you to input a string of characters after the current line. The syntax is

`Input` *`string`*

where:

`string`

Is the text you want placed after the current line.

Deleting and Recovering Deleted Text

The following commands delete or recover deleted text:

Command Line Commands	Prefix Area Commands
`DELete`	`===D=` `==DD=`
`CDel`	
`RECover`	

==D==

To delete a line, type the letter D in the prefix area of the line to be removed, and press the *Enter* key.

You can also use the syntax

```
==Dn=
```

where:

n

Is the number (up to four digits) of lines to be deleted beginning with the line where the command is issued.

==DD=

To delete a block of lines, enter the letters DD in the prefix area in the first and last lines of the block to be deleted. For example:

```
 FOCEXEC(EXAMPLE)                        SIZE=5     LINE=0

===== * * * TOP OF FILE * * *
==DD= THIS LINE WILL BE DELETED, ALONG WITH THE FOLLOWING
===== TWO LINES
==DD= THIS ONE TOO.
=====
=====
===== * * * END OF FILE * * *



====>
                                                EDITING MODE
```

After you use the DD prefix area command, the previous screen looks like this:

```
 FOCEXEC(EXAMPLE)                        SIZE=2     LINE=0

===== * * * TOP OF FILE * * *
=====
=====
===== * * * END OF FILE * * *



====>
                                                EDITING MODE
```

DELete

To delete lines beginning with the line after the current line, use the DELETE command in one of the following forms

```
DELete n
```

where:

n

Is the number of lines to be deleted.

or

```
DELete /text
```

where:

text

All of the lines from the current line to the line with "text" are deleted. "Text" must be preceded by a delimiter, which can be any special character (not alphabetical or numeric) that does not appear in the string itself. In this case, the slash is the delimiter.

CDel

To delete a line that is not the current line, type CD on the command line, position the cursor at the desired line, and press *Enter*.

RECover

Suppose that after making a deletion, you wish to recover the deleted text. Use the RECOVER command, followed by the number of lines to be recovered. The syntax is

```
RECover n
```

where:

n

Is the number of lines to be recovered. Instead of a number, you can use an asterisk (*) to recover all the lines. If *n* is omitted, it defaults to 1.

Note:

- You can only recover the last block of text deleted during your current TED session. After you terminate the session, the text is no longer recoverable.
- The last recovered line becomes the current line.

The following screens illustrate the RECOVER command:

```
 FOCEXEC(EXAMPLE)                      SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
===== THIS SCREEN WILL SHOW WHAT HAPPENS WHEN YOU USE THE
===== RECOVER COMMAND. THE THIRD LINE WILL BE DELETED.
==D== THIS IS THE THIRD LINE.
===== THEN IT WILL BE RETURNED BACK AT THE CURRENT LINE.
===== * * * END OF FILE * * *


====>
                                              EDITING MODE
```

After you press the *Enter* key (and the line is deleted), you can issue the RECOVER command. After you issue this command, the screen appears with the recovered line immediately after the current line.

```
 FOCEXEC(EXAMPLE)                      SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
===== THIS IS THE THIRD LINE.
===== THIS SCREEN WILL SHOW WHAT HAPPENS WHEN YOU USE THE
===== RECOVER COMMAND. THE THIRD LINE WILL BE DELETED.
===== THEN IT WILL BE RETURNED BACK AT THE CURRENT LINE.
===== * * * END OF FILE * * *


====>
                                              EDITING MODE
```

Moving Through a File

Scrolling a screen is like turning the pages of a book. When you move the screen forward or backward, you automatically change the current line. The following commands enable you to scroll through a file:

Command Line Commands	Function Keys
BAckward	PF7 and PF19
FOrward	PF8 and PF20

Command Line Commands	Function Keys
`Top`	
`Bottom`	
`DOWN`	
`UP`	
`NEXT`	
`:n`	
`±n`	

BAckward

The BACKWARD command scrolls the screen toward the beginning of the file. The syntax is

`BAckward` *n*

where:

n

Is the number of screen pages.

FOrward

The FORWARD command scrolls the screen toward the end of the file. The syntax is

`FOrward` *n*

where:

n

Is the number of screen pages.

PF7, PF8, PF19, PF20

Another way to move backward and forward in a file is using the following control keys:

PF7 — Scrolls the screen view back one full screen page. (You can also use PF19.)

PF8	Scrolls the screen view forward one full screen page. (You can also use PF20.)

Top, Bottom

To scroll directly to the top of a file, enter:

```
Top
```

To scroll to the bottom of a file, enter:

```
Bottom
```

DOWN, UP, NEXT

Suppose that you want to move the file up or down a few lines instead of a whole screen. With the DOWN command, you can specify how many lines you want to scroll down. The syntax is

```
DOWN n
```

where:

n

Is the number of lines you want to scroll down.

The NEXT command is identical to DOWN.

With the UP command, you can specify how many lines you want to scroll up. The syntax is

```
UP n
```

where:

n

Is the number of lines you want to move up.

:n

To scroll to a specific line, enter a colon command at the command line, using the following syntax

```
:n
```

where:

n

Is the number of the line to which you want to scroll.

±n

Enter a number preceded by a plus sign to scroll forward or a number preceded by minus sign to scroll backward *n* number of lines.

Locating and Changing Text

When viewing a file that you wish to modify, you can either move the cursor to the lines to be edited and type over the text, or use the LOCATE and CHANGE commands.

Command Line Commands

```
Locate
Change
```

Locate

The LOCATE command searches the file beginning at the current line for a character string you specify. If the character string is located, the line containing the string becomes the current line. The syntax is

```
Locate/string/
```

where:

string

Is the string you wish to locate. The string must have delimiters. You can use a slash (/) or any special character (non-alphanumeric) that does not appear in the string itself. Note that the word LOCATE is optional. You can start with /.

If the string that you seek is behind the current line (toward the top of the file), you can specify a backward search by typing a minus sign (-) in front of the string. For example:

```
LOCATE -/GOOD/
```

To locate more than one occurrence of a string, attach an ampersand (&) to LOCATE (the & command is explained in *Displaying or Repeating the Previous Command* on page 68). For example:

```
&LOCATE/string/
```

Each time you press the *Enter* key, the next string occurrence located appears as the current line.

Change

To change a string of characters at the current line or throughout a file, you can use the CHANGE command. The full syntax of this command is

```
Change/oldstring/newstring/ n m
```

where:

oldstring

Is the sequence of characters that you wish to change.

newstring

Is the new character string.

n

Is the number of lines from the current line that you want to scan and change. You can use an asterisk (*) to indicate all lines in a file from the current line.

m

Is the number of changes on each line. You can use an asterisk (*) to indicate all occurrences in each line.

newstring and *oldstring* must have delimiters. You can use any special character (no alphabetics or numerics) that does not appear in the string itself.

For example:

```
 FOCEXEC(EXAMPLE)                        SIZE=6     LINE=0

===== * * * TOP OF FILE * * *
===== THIS SCREEN SHOWS HOW TO CHANGE A STRING OF CHARACTERS.
===== THIS SCREEN ALSO SHOWS HOW TO USE THE CHANGE COMMAND.
===== NOTICE HOW THE FIRST TWO LINES BEGIN WITH 'THIS SCREEN.'
===== NOTICE HOW EVERYTHING WILL CHANGE FROM THE CURRENT LINE
===== TO THE END OF FILE.
===== THIS SCREEN SHOWS HOW TO CHANGE A STRING OF CHARACTERS.
===== * * * END OF FILE * * *


====> CHANGE/THIS SCREEN/THIS EXAMPLE/* *
                                                 EDITING MODE
```

After you press the *Enter* key, each occurrence of THIS SCREEN changes to THIS EXAMPLE. When you use the CHANGE command, TED displays the number of occurrences changed, as shown below.

```
 FOCEXEC(EXAMPLE)                        SIZE=6     LINE=0

===== * * * TOP OF FILE * * *
===== THIS EXAMPLE SHOWS HOW TO CHANGE A STRING OF CHARACTERS.
===== THIS EXAMPLE ALSO SHOWS HOW TO USE THE CHANGE COMMAND.
===== NOTICE HOW THE FIRST TWO LINES BEGIN WITH 'THIS EXAMPLE.'
===== NOTICE HOW EVERYTHING WILL CHANGE FROM THE CURRENT LINE
===== TO THE END OF FILE.
===== THIS EXAMPLE SHOWS HOW TO CHANGE A STRING OF CHARACTERS.
===== * * * END OF FILE * * *



====>
                                                 EDITING MODE
```

Note that the last line changed has become the current line.

Copying and Moving Text

The following commands duplicate and move text in a file:

Command Line Commands	Prefix Area Commands
COpy	==C== ==CC= ==" "=

Command Line Commands	Prefix Area Commands
DUplicat	==″n=
MOve	==M== ==MM=

COpy

To copy text lines in a file, use the COPY command with the following syntax

```
COpy n m
```

where:

n

Is the number of lines to copy beginning with the current line.

m

Indicates where you want the copied lines placed, as the number of lines away from the current line (relative line position).

For example, if the current line is line 5 and you entered

```
CO 3 10
```

three lines (starting with the current line) would be copied and placed immediately after line 15 (line 5 + 10 lines = line 15).

==C==

To duplicate a line in EDIT mode, enter the letter C in the prefix area. You must then indicate where the copied line will be inserted. Enter either the letter F (following) or P (preceding) in the prefix area of another line. You can also place a number after C to indicate the number of lines you want copied.

==CC=, =="="=

To copy a block of text consisting of more than one line, enter the letters CC in the prefix area of the first and last lines of the block to be copied. Enter either the letter F (following) or P (preceding) in the prefix area of another line depending on where you want the duplicated line(s) to appear. For example:

```
 FOCEXEC(EXAMPLE)                        SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
==CC= THIS EXAMPLE SHOWS HOW TO COPY A BLOCK OF TEXT.
==CC= THIS EXAMPLE ALSO SHOWS HOW YOU TO USE THE CC COMMAND.
===== YOU DON'T HAVE TO READ THIS.
==F== YOU DON'T HAVE TO READ THIS EITHER.
===== * * * END OF FILE * * *


====>
                                                    EDITING MODE
```

When you press the *Enter* key, the following screen appears:

```
 FOCEXEC(EXAMPLE)                        SIZE=6     LINE=0

===== * * * TOP OF FILE * * *
===== THIS EXAMPLE SHOWS HOW TO COPY A BLOCK OF TEXT.
===== THIS EXAMPLE ALSO SHOWS HOW YOU TO USE THE CC COMMAND.
===== YOU DON'T HAVE TO READ THIS.
===== YOU DON'T HAVE TO READ THIS EITHER.
===== THIS EXAMPLE SHOWS HOW TO COPY A BLOCK OF TEXT.
===== THIS EXAMPLE ALSO SHOWS HOW YOU TO USE THE CC COMMAND.
===== * * * END OF FILE * * *



====>
                                                    EDITING MODE
```

Note: F or P may not lie within the block to be copied.

If you place the prefix area command =" "== at the top line and bottom line of a block of text and press *Enter*, the block is duplicated immediately after its present position.

==\"n=

To duplicate a line, enter a double quotation mark followed by the number of times you want the line duplicated. To duplicate a block of text, enter an additional double quotation mark in the prefix area of the last line of the block. If a number (n) is omitted, one line is duplicated. Text appears in lines following the current line.

DUplicat

To duplicate text from the current line to a specified line, use the following syntax

```
DUplicat n m
```

where:

n

Is the number of duplications.

m

Indicates how many lines are included in the duplication.

MOve

To move one or more lines of text, use the MOVE command with the following syntax

```
MOve n m
```

where:

n

Is the number of lines you want moved, starting with the current line.

m

Indicates how many lines down from the current line (relative line position) you want the moved lines placed.

==M==

To move a line, enter the letter M in the prefix area. Then indicate where the moved line will be inserted. Enter either the letter F (following) or P (preceding) in the prefix area of another line depending on where you want the line(s) to be placed. You can also place a number next to M, indicating the number of lines you want moved.

=MM==

To move a block of text, enter the letters MM in the prefix area of the first and last lines of the block to be moved. Then indicate where the moved lines will be inserted. Enter either the letter F (following) or P (preceding) in the prefix area of another line depending on where you want the line(s) to be placed. Note that F or P may not lie within the block to be moved.

Joining and Splitting Text

In addition to moving or copying text in a file, you can join, move, or split lines using the following commands:

Command Line Commands	Prefix Area Commands
Join	==J==
SPLit	==SP=

==J==

To join two consecutive lines, enter the letter J in the prefix area of the line that will be joined. Then position the cursor on the spot where you want the join to take place and press the *Enter* key.

```
 FOCEXEC(EXAMPLE)                      SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
===== THIS SCREEN SHOWS HOW THE ==J== COMMAND WORKS.
==J== THIS LINE WILL BE JOINED
===== TO THIS LINE.
===== JUST POSITION THE CURSOR WHERE YOU WANT TO JOIN LINES.
===== * * * END OF FILE * * *


====>
                                                    EDITING MODE
```

Note that the cursor is at the end of the line. When you press the *Enter* key, the following screen appears:

```
 FOCEXEC(EXAMPLE)                      SIZE=3     LINE=0

===== * * * TOP OF FILE * * *
===== THIS SCREEN SHOWS HOW THE ==J== COMMAND WORKS.
===== THIS LINE WILL BE JOINED TO THIS LINE.
===== JUST POSITION THE CURSOR WHERE YOU WANT TO JOIN LINES.
===== * * * END OF FILE * * *



====>
                                                    EDITING MODE
```

Join

To join two consecutive lines from the command line, type the Join command, position the cursor at the place where you want the join to take place, and press the *Enter* key.

=SP==

To split a line, enter the letters SP in the prefix area, place the cursor where the text is to be split into a separate line, and press the *Enter* key.

SPLit

You can also use the SPLIT command to split a line after the cursor position and create a new line. Type the command at the command line, position the cursor where the text is to be split, and press the *Enter* key.

Editing Multiple Files

By entering any of the following commands at the command line you can display, edit, or create up to four files at the same time (or four sections of the same file):

Command Line Commands

```
SPH
SPLITH
SPV
SPLITV
TEd
```

Each file remains on the screen until you enter a FILE or QUIT command. You can use any TED facility in each window. To move the cursor from one window (that is, file) to another, use the cursor control keys.

SPH, SPLITH

To split the screen horizontally and call a new file or an existing one, use the following syntax

```
SPH [filename]
SPLITH [filename]
```

where:

filename

Is the name of the file you want displayed horizontally. If you omit the file name, another copy of the current file is displayed.

The command SPLITH is identical to SPH. For example, if you enter SPLITH with no file name, the existing file is repeated in a second, horizontal, window of the screen, as shown below:

```
 FOCEXEC(EXAMPLE)                      SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
===== THIS IS AN EXAMPLE OF SPLIT SCREEN IN TED.
===== YOU CAN USE SPH, SPLITH, SPV, OR SPLITV COMMANDS.
===== IF YOU DO NOT SPECIFY A FILENAME, THE FILE PRESENTLY
===== LOADED IN TED WILL BE SPLIT.
===== * * * END OF FILE * * *

====>
                                                              EDITING MODE
--------------------------------------------------------------------------
FOCEXEC(EXAMPLE)                     SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
===== THIS IS AN EXAMPLE OF SPLIT SCREEN IN TED.
===== YOU CAN USE SPH, SPLITH, SPV, OR SPLITV COMMANDS.
===== IF YOU DO NOT SPECIFY A FILENAME, THE FILE PRESENTLY
===== LOADED IN TED WILL BE SPLIT.
===== * * * END OF FILE * * *

====>
```

SPV, SPLITV

To split the screen vertically and call a new or existing file, use the following syntax

```
SPV [filename]
SPLITV [filename]
```

where:

filename

Is the name of the file you want displayed vertically. If you omit the file name, another copy of the current file is displayed.

The command SPLITV is identical to SPV.

TEd

To edit another file without using the split screen facility, use the following syntax on z/OS:

```
TED ddname(member)
```

To edit another file without using the split screen facility, use the following syntax on distributed systems:

```
TED filename.ext
```

where:

ddname

Is the DDNAME allocated to the data set that contains the member to be edited on z/OS.

member

Is the name of the new member to be edited or created. Entering TED without the file name proceeds to the next file in the current window on z/OS.

filename.ext

Is the file name and extension of the file to be edited or created on distributed systems.

Transferring Text Between Files and Temporary Storage

To insert all or part of one file into another file, use the following commands:

Command Line Commands	Prefix Area Commands
PUT	==PP=

Command Line Commands	Prefix Area Commands
PPUT	==PL=
PUTD PPUTD	==PD=
Get	===G=

==PP=, PUT, PPUT

To temporarily store a copy of a line, or block of lines for subsequent insertion in the same or another file, enter the letters PP in the first and last lines to be transferred. The lines remain in the source file.

You can also use the command PUT using the following syntax

```
PUT [n] [filename]
```

where:

n

Is the number of lines to pick up starting from the current line. The default is 1.

filename

Is the *ddname(membername)* of the file where you want to store the lines of text on z/OS. On distributed systems, it is a file name (*name.ext*). If you omit the file name, it defaults to a temporary storage area. You can retrieve the lines using the GET or ==G== command.

For example:

```
 FOCEXEC(EXAMPLE)                      SIZE=4     LINE=0

===== * * * TOP OF FILE * * *
==PP= THIS IS AN EXAMPLE OF COPYING TEXT FROM ONE FILE
===== TO ANOTHER. ==PP= COPIES THE SPECIFIED TEXT AND
===== PUTS IT IN A TEMPORARY FILE. YOU CAN THEN USE THE
==PP= GET COMMAND TO RETRIEVE THE COPIED TEXT.
===== * * * END OF FILE * * *



====>
                                                EDITING MODE
```

If the file name already exists, and you want to overwrite the existing file, use the command:

```
PPUT
```

==PLn

To insert *n* lines of text into a temporary file, use the PL prefix-area command. When no *n* is specified, it defaults to 1. The line remains in the source file.

==PD=, PUTD, PPUTD

To temporarily store a block of lines and delete them from the source file, enter the letters PD in the prefix area of the first and last lines of the block of text. The command PUTD n has the same effect as ==PD=. It has the following syntax

```
PUTD n [filename]
```

where:

n

Is the number of lines to pick up and delete (from the source file) beginning with the current line.

filename

Is the *ddname(member name)* of the file where you want to store the lines of text on z/OS. On distributed systems, it is a file name (*name.ext*). If you omit the file name, it defaults to a temporary storage area. You can retrieve the lines using the GET or ==G== command.

If the file name already exists, and you want to overwrite the existing file, use the command:

```
PPUTD
```

==G==, Get

To recall lines from the default temporary storage file, enter the letter "G" in the prefix area of the line preceding the point of insertion. For example:

```
 FOCEXEC(EXAMPLE2)                     SIZE=2     LINE=0

===== * * * TOP OF FILE * * *
===== THIS IS A NEW FILE IN WHICH COPIED TEXT FROM ANOTHER
==G== FILE WILL BE INSERTED BELOW USING ===G= command.
===== * * * END OF FILE * * *



====>
                                              EDITING MODE
```

When you press the *Enter* key, the following screen appears:

```
 FOCEXEC(EXAMPLE2)                     SIZE=6     LINE=0

===== * * * TOP OF FILE * * *
===== THIS IS A NEW FILE IN WHICH COPIED TEXT FROM ANOTHER
===== FILE WILL BE INSERTED BELOW USING ===G= command.
===== THIS IS AN EXAMPLE OF COPYING TEXT FROM ONE FILE
===== TO ANOTHER. ==PP= COPIES THE SPECIFIED TEXT AND
===== PUTS IT IN A TEMPORARY FILE. YOU CAN THEN USE THE
===== GET COMMAND TO RETRIEVE THE COPIED TEXT.
===== * * * END OF FILE * * *


====>
                                              EDITING MODE
```

You can also use the command GET with the following syntax

```
Get [filename]
```

where:

filename

Is the *ddname(membername)* of the file that contains the text on z/OS. On distributed systems, it is a file name (*name.ext*). If you omit the file name, TED searches for text in the temporary storage area used by PUT.

Note: When transferring lines between files using temporary storage, do not leave the TED environment. Rather, follow this procedure:

1. Place the lines in temporary storage using one of the PUT or PUTD commands described above.
2. Enter the target file, using the TED command as described in *Editing Multiple Files* on page 60.
3. Retrieve the lines from temporary storage using the GET command.

Displaying a Scale and Line Numbers

To display or cancel a scale or line numbers on the screen, use the following commands:

Command Line Commands

```
NUm ON
NUm OFF
SCale ON
SCale OFF
```

NUm ON

To replace the prefix area with numbers, enter

```
NUm ON
```

at the command line. Prefix area commands can be issued while line numbers are displayed. If you use this command in TYPE mode, it changes to EDIT and displays the line numbers. For example:

```
 FOCEXEC(EXAMPLE)                      SIZE=6     LINE=0

00000 * * * TOP OF FILE * * *
00001 THIS SCREEN SHOWS HOW
00002 LINE NUMBERS ARE DISPLAYED
00003
00004
00005
00006
00007 * * * END OF FILE * * *


====>
                                                EDITING MODE
```

NUm OFF

To replace the numbers with =, issue

```
NUm OFF
```

at the command line.

Note: This command returns you to EDIT.

SCale ON

To display a scale on the screen, type

```
SCale ON
```

at the command line, as seen in the following:

```
 FOCEXEC(EXAMPLE)                      SIZE=6     LINE=0
      ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
00000 * * * TOP OF FILE * * *
00001 THIS SCREEN SHOWS A SCALE
00002
00003
00004
00005
00006
00007 * * * END OF FILE * * *


====>
                                              EDITING MODE
```

SCale OFF

To remove the scale on the screen, type

```
SCale OFF
```

at the command line.

Displaying or Repeating the Previous Command

The following commands enable you to display or repeat the previous command:

Command Line Commands	Function Keys
=	PF5
?	PF6
&	

=, PF5

You can repeat the last command entered by typing the equal sign at the command line or pressing the PF5 key.

?, PF6

To display the previous command, enter ? at the command line, or press the PF6 key.

&

If you wish to repeat a command, precede it with & on the command line and press *Enter* again.

Moving the Screen Display

In TED, you may create lines of up to 159 characters. When a line of text exceeds the 80 columns on a screen, you can move the screen display left or right to view the additional text.

Command Line Commands	Function Keys
RIght RIGHTP	PF11, PF23
LEft LEFTP	PF10

RIght

To move one full screen view (80 columns) to the right, enter

```
RIght
```

at the command line. You can also follow this command with a number. This will move that number of columns to the right.

RIGHTP, PF11

To move 30 columns to the right, press the PF11 key, or use the following syntax:

```
RIGHTP
```

LEft

To move one full screen (80 columns) to the left, enter

```
LEft [n]
```

at the command line. You can also follow this command with a number. This will move the screen that number of columns to the left.

LEFTP, PF10

To move 30 columns to the left, press the PF10 key, or use the following syntax:

```
LEFTP
```

Specifying Uppercase and Lowercase Text

The following commands control uppercase or lowercase text in a file.

Command Line Commands

```
CAse M
CAse U
UPPercas
LOwercas
```

CAse M

To have uppercase and lowercase characters enter

```
CAse M
```

at the command line.

Note: You must issue this command before entering the text because the default is uppercase.

CAse U

To get only uppercase characters enter:

```
CAse U
```

Note that although characters may be typed in lowercase, they will be converted to uppercase when you press the *Enter* key.

UPPercas

The UPPERCAS command sets the text to uppercase from the current line to a target line. The syntax is

```
UPPercas n
```

where:

n

Is the number of lines to be converted to uppercase starting with the current line.

LOwercas

The LOWERCAS command sets the text to lowercase from the current line to a target line. The syntax is

```
LOwercas n
```

where:

n

Is the number of lines you want to be lowercase starting with the current line.

Ending a TED Session

The following commands are used to terminate a TED session. With the exception of SAVE, which keeps you in TED, each of these commands returns you to the FOCUS command level.

Command Line Commands	Function Keys
Quit	PF3
QQuit	
FILe	

Command Line Commands	Function Keys
SAve	
FFILE	
SSAVE	

Quit, PF3

To terminate a TED session when the file has not been changed, enter

```
Quit
```

at the command line, or use the PF3 key.

QQuit

To terminate a TED session after making changes to a file that you do not wish to save, enter

```
QQuit
```

at the command line.

FILe

To terminate a TED session and save the changed file, enter

```
FILe [filename]
```

where:

filename

Is the name of the saved file. The default is the file name that appears on the first line of the screen.

For example:

```
 FOCEXEC(EXAMPLE)                        SIZE=5     LINE=0
===== * * * TOP OF FILE * * *
===== THE INPUT MODE IS AN EASY WAY
===== TO ENTER DATA. SIMPLY TYPE THE DATA,
===== AND PRESS THE TAB KEY TO GO TO THE NEXT LINE.
===== WHEN YOU HAVE FINISHED, JUST PRESS THE ENTER KEY TWICE,
===== AND YOU WILL BE BACK IN TYPE MODE.
===== * * * END OF FILE * * *




====> FILE MASTER(EXAMPLE
                                              EDITING MODE
```

SAve

You can also use the SAVE command to save a file as it appears and continue the TED session using the following syntax

```
SAve [filename]
```

where:

filename

Is the name of the saved file. The default is the file name that appears on the first line of the screen.

FFILE, SSAVE

There are times when you wish to store a file under a name other than the original name. To do so, see the TED command FILE. If the new file name already exists, you will be informed and asked to use either FFILE or SSAVE to overwrite the existing file.

Note: FFILE returns you to the FOCUS command level; SSAVE enables you to continue the TED session.

Accessing the HELP File

While in TED, you can enter the command HELP on the command line to display a list of PF key functions. Entering HELP again displays a file containing explanations of all TED commands.

You may also press the PF1 key to display a list of PF key functions. If you press the PF1 key again, the HELP file is displayed.

Note: Most TED commands (for example, LOCATE) are accessible while the HELP file is displayed.

Editing FOCEXECs

You can create FOCUS executable procedures (FOCEXECs) using TED, just as you can create any other file. One advantage of creating or editing FOCEXECs in TED is that you can use the RUN command, which enables you to run a FOCEXEC from within TED without moving to an editor outside the FOCUS command level. For example:

```
 FOCEXEC(M1)                              SIZE=10    LINE=0

00000 * * * TOP OF FILE * * *
00001 MODIFY FILE EMPLOYEE
00002 FREEFORM EMP_ID CURR_SAL
00003 MATCH EMP_ID
00004  ON NOMATCH REJECT
00005  ON MATCH UPDATE CURR_SAL
00006 DATA
00007 EMP_ID=071382660, CURR_SAL=21400.50, $
00008 EMP_ID=112847612, CURR_SAL=20350.00, $
00009 EMP_ID=117593129, CURR_SAL=22600.34, $
00010 END
00011 * * * END OF FILE * * *

====> RUN
                                                      EDITING MODE
```

Once you type RUN and press the *Enter* key, this FOCEXEC is executed by FOCUS. Also note that you can add parameters to the RUN command. For example:

```
====> RUN ECHO=ON
```

If there is an error in the FOCEXEC, simply enter

```
TED
```

after the error message. The file will be redisplayed in TED, with the error line as the current line.

In the following example, the FOCEXEC has an incorrect field name (EMPID, should be EMP_ID):

```
 FOCEXEC(M2)                            SIZE=9     LINE=0

00000 * * * TOP OF FILE * * *
00001 MODIFY FILE EMPLOYEE
00002 FREEFORM EMPID CURR_SAL
00003 MATCH EMP_ID
00004  ON NOMATCH REJECT
00005  ON MATCH UPDATE CURR_SAL
00006 DATA
00007 EMP_ID=071382660, CURR_SAL=21400.50, $
00008 EMP_ID=112847612, CURR_SAL=20350.00, $
00009 EMP_ID=117593129, CURR_SAL=22600.34, $
00010 END
00011 * * * END OF FILE * * *

====>
                                                      EDITING MODE
```

After you type RUN and press the *Enter* key, FOCUS displays the following error:

```
 ERROR AT OR NEAR LINE      2  IN PROCEDURE M2      FOCEXEC *
(FOC419) FIXFORM SUBCOMMAND ELEMENT OR FIELDNAME NOT RECOGNIZED: EMPID
 BYPASSING TO END OF COMMAND
```

If you issue the TED command (it is not necessary to include the file name), you are placed in EDIT mode with the following screen displayed. The current line is FREEFORM EMPID CURR_SAL

```
 FOCEXEC(M2)                            SIZE=10    LINE=2

00002 FREEFORM EMPID CURR_SAL
00003 MATCH EMP_ID
00004  ON NOMATCH REJECT
00005  ON MATCH UPDATE CURR_SAL
00006 DATA
00007 EMP_ID=071382660, CURR_SAL=21400.50, $
00008 EMP_ID=112847612, CURR_SAL=20350.00, $
00009 EMP_ID=117593129, CURR_SAL=22600.34, $
00010 END
00011 * * * END OF FILE * * *

====>
                                                      EDITING MODE
```

Note: FOCEXECs are described in detail in the *Developing Applications* manual.

Personalizing TED: PROFILE and PFnn

There are editing features you may wish to use every time you enter TED, (such as NUM ON, CASE M, CURLINE, or EDIT). You can establish these in a profile that is automatically executed every time you enter TED before the first screen appears. These commands will remain in effect for the duration of your TED session, unless you change them.

On z/OS, the file name must be

```
FOCEXEC(TEDPROF)
```

where:

`TEDPROF`

Is a member of the PDS allocated to DDNAME FOCEXEC.

On distributed systems, the file name must be

```
tedprof.fex
```

In addition to creating a profile, you can define function keys by issuing the following command at the TED command line

PF*nncommand*

where:

nn

Is the number of the PF key.

command

Is the new function of the PF key. Note that there is a limit of 40 characters.

For example

```
PF13 DELETE
```

defines the PF13 key to perform the DELETE command.

Note: TED cannot manipulate files from the PROFILE. Commands such as GET in PROFILE TED cause an error.

Syntax Summary

There are four environments in TED. Access these environments by issuing the following commands:

```
TYPE
EDIT
INPUT
PAINT
```

Function Keys

The following list shows the PF Key assignments in TED.

Key	Action
PF1, PF13	Displays the meaning of the keys and help information.
PF2	Inserts a line after the cursor.
PF3, PF15	QUITs.
PF4	PAINTs.
PF5	REPEATs last command.
PF6	RECALLs last command.
PF7, PF19	Moves BACKWARD one full screen.
PF8, PF20	Moves FORWARD one full screen.
PF10	Moves to the LEFT one page.
PF11, PF23	Moves to the RIGHT one page.
PF22	Moves LEFT one character.

Prefix-Area Commands

The following commands can be issued in the prefix area of a TED file. To execute one, place the appropriate characters anywhere in the prefix area and press *Enter*.

Command	Action
`==/==`	Becomes current line.
`==DD=`	Deletes block. Requires start and end lines.

Command	Action
==D*n*=	Deletes *n* lines.
==MM=	Moves block.
==I*n*=	Inserts *n* lines.
==CC=	Copies block. Requires start and end lines.
==A*n*=	Inserts *n* lines.
==PP=	Puts block into stack. Requires start and end lines.
==*"n*=	Duplicates *n* times.
==""=	Duplicates block. Requires start and end lines.
==M*n*=	Moves *n* lines.
==SP=	Splits line (at cursor).
==C*n*=	Copies *n* lines.
==J==	Joins line (at cursor).
==P*n*=	Puts *n* lines into a temporary file.
==PD=	Stores a block of lines in a temporary file and deletes it from a source file.
==PL*n*	Puts *n* lines into stack.
==G==	Gets lines from stack.

Command Line Commands

These commands may be executed from the TED command line.

Note that uppercase letters in the following list indicate the shortest acceptable abbreviations.

Any command that is preceded by & remains on the command line and is not erased when the *Enter* key is pressed.

Command	Action
`Add` *n*	Adds *n* lines after current line.
`BAckward` *n*	Moves backward *n* pages.
`Bottom`	Goes to bottom of file.
`CAse` *m/u*	Displays mixed upper/lowercase, uppercase.
`CDel`	Deletes line pointed to by cursor.
`Change /old/new/`*n m*	Changes old to new *n* times on *m* lines (or * *).
`CINS`	Inserts line after cursor.
`COpy` *target1 target2*	Copies from current line through *target1* after *target2*.
`CUrline` *n*	Sets current line to specified line number.
`DElete/`*target text*	Deletes from current line up to the line with target text.
`DELete` *n*	Deletes *n* number of lines.
`DOwn` *n*	Moves forward *n* lines.
`DUplicat` *n*	Targets duplicates from current line until target *n* times.
`Edit`	Displays mode with five-character prefix area.
`FIle` *ddname(member)* `FIle` *name.ext*	On z/OS, saves file as *member*. On distributed systems, saves file as *name.ext*.
`FILEName` *ddname (newmember)* `FILEName` *filename.ext*	On z/OS, changes the default member used for FILE and SAVE commands. On distributed systems, changes the default file name used for FILE and SAVE commands.
`FOrward` *n*	Moves forward *n* pages.

Command	Action
`Get ddname(member)` `Get name.ext`	On z/OS, gets a member or gets stack if DDNAME and member are not specified. On distributed systems, gets a file or gets stack if the file name is not specified.
`Help`	Retrieves the HELP file.
`Input string`	Inserts line after current line.
`Join`	Joins line after cursor to cursor position.
`LEft n`	Moves one full screen to left *n* columns.
`LEFTP`	Moves one half screen to left.
`Locate/string/`	Locates a string, search forwards.
`LOWercas target`	Sets print to lowercase from current line to target line.
`MOve target1 target2`	Moves block from current line to *target1*, after *target2*.
`MVS command`	Issues TSO command.
`Next n`	Moves forward *n* lines.
`NUmber ON/OFF`	Sets up prefix area with numbers.
`Overlay string`	Overlays string on current line; existing text remains after end of string.
`PAINT n`	Paints the "nth" CRTFORM.
`PFnn string`	Sets PF key *nn* to the specified string.
`Put n [filename]`	Puts *n* lines to specified file.
`PUTD n [filename]`	See PUT, but lines are deleted.
`QQuit`	Quits even if changes have been made. Changes are not recorded.

Command	Action
`Quit`	Quits if no changes have been made. Changes are not recorded.
`RECover` *`n`*	Recovers lines that were just deleted.
`Replace` *`string`*	Writes string on current line instead of existing text.
`RESet`	Resets to original mode; cancels pending prefix operations.
`RIght` *`n`*	Moves one full screen to the right or *n* columns.
`RIGHTP`	Moves half a screen to the right.
`RUn` *`parameter`*	Files and executes FOCEXEC that is being edited along with the specified parameters.
`SAve` *`filename`*	Saves file.
`SCale ON/OFF`	Displays a scale at the top of the screen.
`SPlit`	Splits line at cursor position and create a new line.
`SPH [`*`filename`*`]`	Splits screen horizontally.
`SPLITH [`*`filename`*`]`	Splits screen horizontally.
`SPLITV [`*`filename`*`]`	Splits screen vertically.
`SPV [`*`filename`*`]`	Splits screen vertically.
`SUBmit`	Submits TED image for batch execution.
`TED` *`ddname`* `TED` *`name.ext`*	Edits another file on z/OS. Edits another file on distributed systems.
`Top`	Goes to top of file.
`TSO` *`command`*	Issues TSO command.
`TYpe`	Sets mode to data display, no prefix area.

Command	Action
`UNIX` *`command`*	Issues a UNIX command.
`Up n`	Moves backward *n* lines.
`UPPercas` *`target`*	Sets print to uppercase from current line to target line.
`- /string/`	Performs a backward search.
`=`	Repeats last command.
`?`	Shows last command.
`? n`	Shows text of error message number *n*.
`?F` *`filename`*	Shows fields in file *filename*.

Chapter 3

Invoking the ISPF Editor on z/OS

The IEDIT command opens the system editor (ISPF EDIT) from within FOCUS.

In this chapter:

- Editing Files With IEDIT
- Installing IEDIT
- Using IEDIT

Editing Files With IEDIT

IEDIT adds the following TED-like functionality to the editing session:

- You can issue the RUN command with or without parameters to save and execute a FOCEXEC that is open in the editor.
- The editor is positioned to the line number where FOCUS encounters an error while executing a FOCEXEC.
- The last FOCEXEC executed opens when no file name is specified when you issue the IEDIT command.

All system editor commands are valid, and any editor environment you establish as your default should be in force. TED commands other than RUN are not valid.

Syntax: How to Edit Files With IEDIT

```
IEDIT [ddname|FOCEXEC]
IEDIT [ddname(member)]
IEDIT [member]
```

where:

ddname

Is the ddname of the file to edit. If no ddname is supplied, it defaults to FOCEXEC.

member

Is the member name of the file to edit in the PDS allocated to the ddname. **Note:** If only one name is specified in the command and it is both a ddname and a member name (in the FOCEXEC library), the file allocated to the ddname is edited.

For example, if the FOCEXEC library has a member named FEX1 and a sequential file is allocated to ddname FEX1, the following command edits the sequential file:

```
EDIT FEX1
```

filename

Is the name of the file to edit. If omitted, it defaults to the name of the last FOCEXEC that was executed.

Reference: Usage Notes for IEDIT

- The TED command and the IEDIT command are not available once you are in the editor.
- IEDIT is not supported under MSO.

Example: Editing a File With IEDIT

To edit a Master File (allocated to DDNAME MASTER) named CENTORD: z/OSz/OS

```
IEDIT MASTER(CENTORD)
```

To edit a FOCEXEC named LOADORD:

```
IEDIT LOADORD
```

To edit the last FOCEXEC that was run:

```
IEDIT
```

Installing IEDIT

Two files are required for using IEDIT. They are distributed as members of the FOCCTL.DATA data set, and must be copied to a library in the concatenation of data sets allocated to DDNAME SYSPROC.

- **FOCIESTL.** This is a macro that enables the RUN command and determines where in the file the editor is positioned.
- **FOCIERUN.** This is a macro that processes the RUN command.

Using IEDIT

The editor is the ISPF editor with the RUN command added.

When you issue the =X command to exit ISPF, you return to FOCUS.

If you split the screen during an editing session, the new session is not part of FOCUS, and does not have the RUN command or other IEDIT features available.

Chapter 4

Terminal Operator Environment

The FOCUS Terminal Operator Environment is an optional window-oriented environment. It is easy to use and provides facilities that increase your productivity.

In this environment, your screen is divided into work areas called windows. Up to seven windows may appear on the screen at once. Each window accepts a specific type of user activity or performs a task.

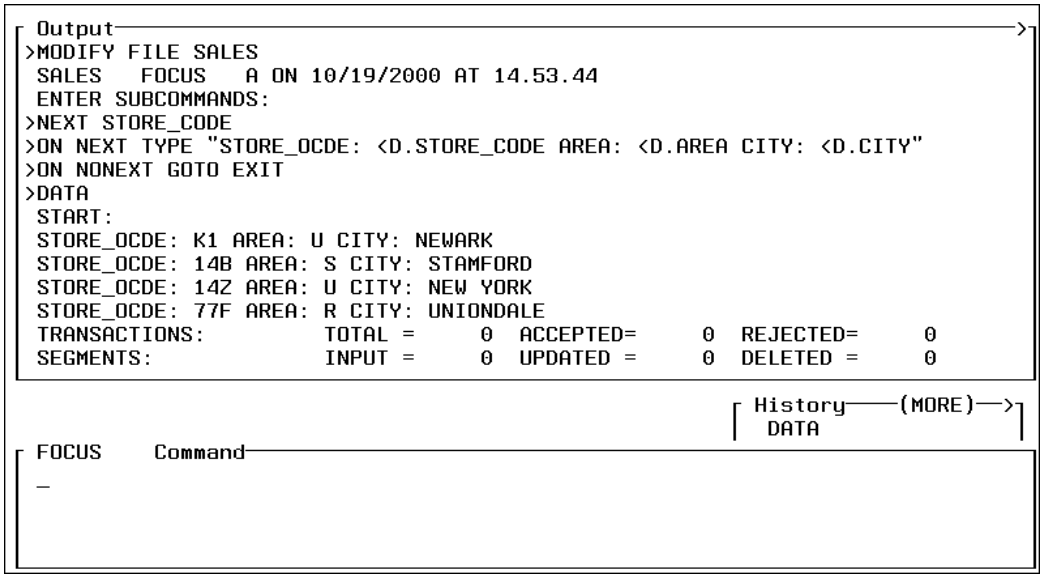
These topics use the SALES and EMPLOYEE data sources in the examples. For more information about either data source, see the *Creating Reports* manual.

In this chapter:

- Illustrating the Terminal Operator Environment
- Invoking the Terminal Operator Environment
- Activating a Window
- Types of Windows
- Window Commands

Illustrating the Terminal Operator Environment

The following sample screen illustrates the Terminal Operator Environment. The blank Command Window is available for commands and requests. The Output Window displays a brief MODIFY procedure as input and the resulting output. The History Window (with the MORE message) has more than one screen of recorded commands and requests. The Table Window retains the most recent TABLE report for the session.



Using predefined program function keys (PF keys), you can move around the screen to activate a window, enlarge a window to full screen size, or scroll window contents. You can also use WINDOW commands to control window behavior and to customize your screen.

Invoking the Terminal Operator Environment

To enter the Terminal Operator Environment, issue the WINDOW ON command from the FOCUS command level. To make the Terminal Operator Environment your FOCUS default environment, include the WINDOW ON command in your PROFILE FOCEXEC. If you have been working in FOCUS, you do not need to reissue USE commands or reset parameter settings for your session.

All FOCUS and operating system commands are available as usual except for interrupt commands like KX, KT, RT, and ?.

Syntax: How to Enter the Terminal Operator Environment

```
WINDOW {ON|OFF}
```

where:

`ON`

Invokes the Terminal Operator Environment.

`OFF`

Enter the command WINDOW OFF from the Command Window to exit the Terminal Operator Environment and return to the FOCUS command level. This value is the default.

After the WINDOW ON command is specified, the Command Window, the Output Window, and the History Window appear on the screen in their default positions. The Command Window is highlighted which indicates that it is activated and ready for commands or requests.

Activating a Window

Although several windows may appear on the screen, only one may be used at a time. This active window appears highlighted. You can use any of the following ways to move the cursor around the screen and to activate a window:

- Move the cursor to the next window using the TAB key or cursor control keys and then press *Enter*.
- From another active window, press *Enter* to return to an active Command Window.
- Press the *PF12* key to move clockwise around the screen.
- Specify a window using the WINDOW ACTIVE command (see *Window Commands* on page 97 for WINDOW commands).

Note: FOCUS automatically activates a window when it is required by the flow of execution. For example, after you execute a report request from the Command Window, the Output Window becomes active and available for scrolling.

Once the window is activated, you may continue to work in it and use the PF keys.

Types of Windows

The Terminal Operator Environment consists of seven types of windows. Each window performs a function or accepts certain activities:

Window	Function
Command	Accepts user input: all FOCUS commands and requests, operating commands, and WINDOW commands (see *Command Window* on page 90).
Output	Displays Command Window input and resulting output; accesses HotScreen facility (see *Output Window* on page 93).
History	Lists commands and requests entered in the Command Window (see *History Window* on page 93).
Help	Displays function key settings. You may redefine settings for the current session (see *Help Window: Revising PF Key Settings* on page 94).
Table	Displays the most recent TABLE request (see *Table Window* on page 95).
Error	Displays FOCUS error messages (see *Error Window* on page 96).

Command Window

All commands and requests are entered in the Command Window. You can enter a FOCUS command exactly as you would at the FOCUS command level. The Command Window accepts up to four lines of text. You can enter any combination of commands and requests.

Type the command in the Command Window, then press *Enter*. The command is copied to the Output Window and to the History Window and is submitted for execution.

Requests are handled in a similar manner. Type your FOCUS TABLE, GRAPH, or MODIFY request. Then, press *Enter*. The request is copied to the Output Window and to the History Window and is submitted for execution.

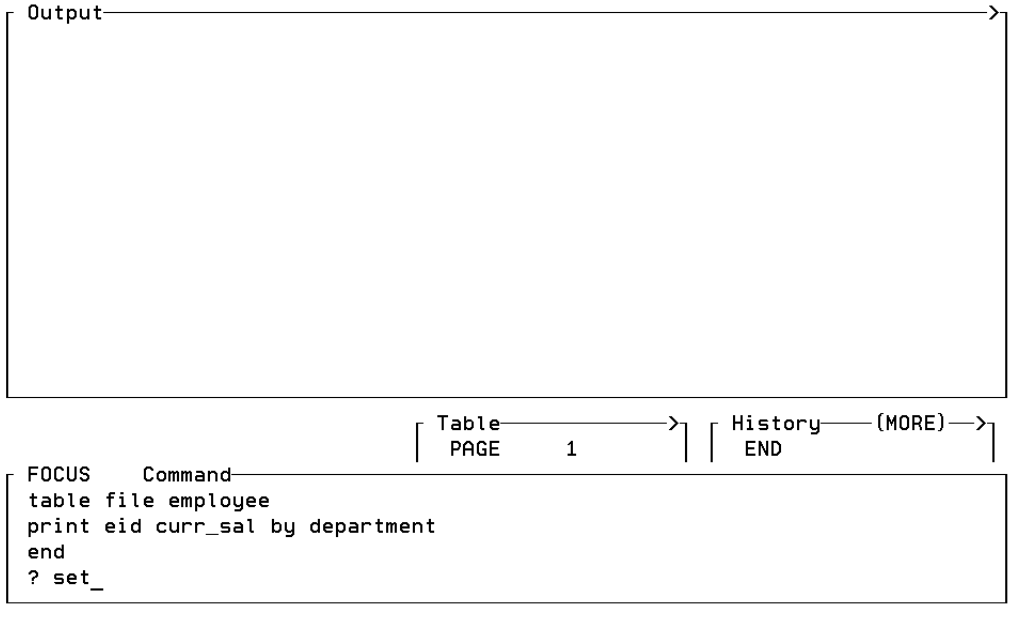
If your request is longer than four lines, press *PF2* to enlarge the window. Finish entering the rest of the request; do not press PF2 again. Press *Enter*. The complete request is copied to the Output Window and to the History Window and is submitted for execution.

If you pressed PF2 and the Command Window returned to its original size, press PF8 and scroll to the end of your request. Now, press *Enter*. This ensures that the entire request, instead of a partial request, is submitted.

Note: If you type over an existing command or request, be sure to delete leftover characters.

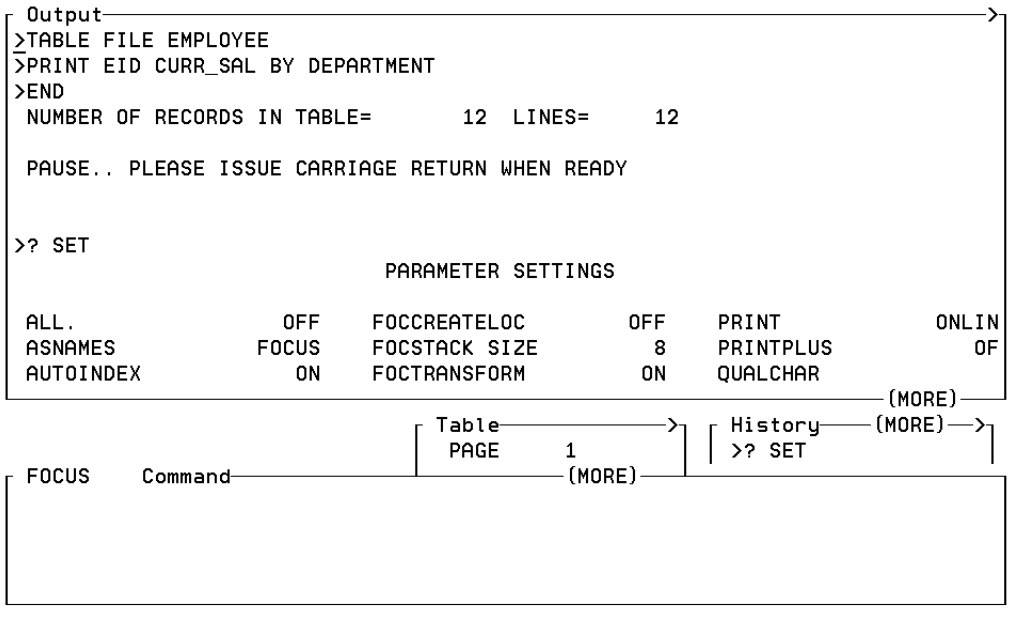
If you enter four FOCUS commands (each on a separate line) or a command and request (totaling four lines), the first item (command or request) is copied to the appropriate windows, submitted, and processed and followed by the next item. The Output and History Windows display each item in succession.

In the sample screen below, the Command Window contains a combination report request and a query command.



After the *Enter* key is pressed, the first item (a request) is copied to the Output and History Windows. The Output Window becomes active and the HotScreen facility displays the report. When you press *Enter* and return from HotScreen to the Output Window, the Output Window displays the query command as input and its output (in this case, parameter settings).

The following sample screen illustrates the Output Window after HotScreen displays the report.



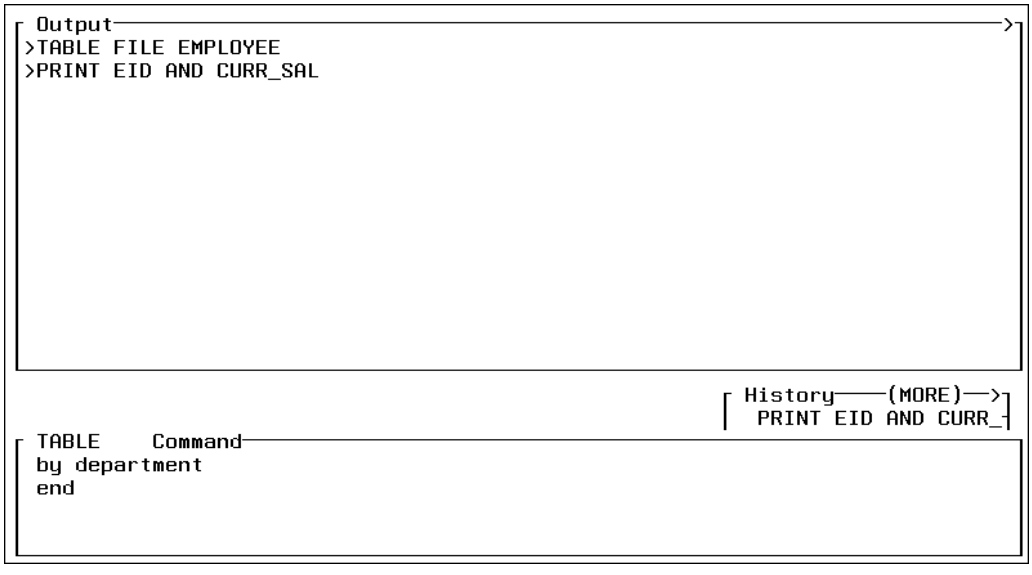
The Command Window also displays the current command mode in a title area. The title area is on the left side of the window's top border. It functions like the prompt does in the default FOCUS environment. In the previous screen sample, for example, the title area displays

```
FOCUS Command
```

because the ? SET query command is a general FOCUS command. If, for example, the last request was a TABLE request or an incomplete request and FOCUS expects another subcommand, the title area displays:

```
TABLE Command
```

In this sample screen, part of the TABLE request has been submitted. The title area indicates a current command mode of TABLE.



Note: To return to the Command Window from another active window, press the *Enter* key.

Output Window

The Output Window functions as a session log. It displays every line of input entered at the Command Window and every resulting line of output. Each line is displayed in the sequence in which it was submitted or generated. Each line of input from the Command Window begins with a caret (>).

Note: The Output Window also accesses the HotScreen facility. After executing a TABLE request, the Output Window becomes active; FOCUS pauses and displays the number of records and lines retrieved. Press *Enter* to display the TABLE report in HotScreen. Press *Enter* again to exit HotScreen and return to the Terminal Operator Environment.

The Output Window does not log the report as displayed in the HotScreen facility; it records only the report request. The Table Window contains the most recent report (see *Table Window* on page 95).

History Window

The History Window provides a history of your FOCUS session. It records commands entered from the Command Window. You can review up to 40 previously typed command lines. For example, you could refer to it to see how you specified an earlier TABLE request.

In the History Window, an asterisk (*) indicates an incorrect command or a mistake in syntax. A caret (>) indicates a command. A request with subcommands is treated as one command and begins with one caret.

You can also recall an old command from the History Window into the Command Window, edit it, and resubmit it. To do so, position your cursor at the command in the History Window and press PF6 or use the WINDOW RECALL command described in *Recalling Commands* on page 105.

Help Window: Revising PF Key Settings

The Help Window displays the program function settings and enables you to change those settings.

To activate the Help Window, press PF1 or enter the WINDOW HELP command from the Command Window. The Help Window overlays existing windows. To deactivate the Help Window and remove it from the screen, press PF1 again.

The default key settings are:

Key	Value
PF1, PF13	Help
PF2, PF14	Zoom
PF3, PF15	
PF4, PF16	Scroll Top
PF5, PF17	Scroll Bottom
PF6, PF18	Recall
PF7, PF19	Scroll Backward
PF8, PF20	Scroll Forward
PF9, PF21	Move Cursor
PF10, PF22	Scroll Left
PF11, PF23	Scroll Right
PF12, PF24	Next

Note: PF3 and PF15 are undefined. You may type a command in the blank next to either key.

To change a key setting for the current session, type a new command over the old command and press *Enter*. For more space to specify a long command, enlarge the Help Window with the PF2 key. Be sure to erase leftover characters.

Example: Erasing the Window Contents

To erase the window contents, specify a Clear key for the Output Window by defining the PF3 key (which happens to be undefined):

```
PF3 CLEAR OUTPUT
```

Example: Assigning WINDOW Commands as Key Settings

When you assign WINDOW commands as key settings, the WINDOW keyword is not required. If you assign FOCUS commands as key settings, the FOCUS keyword is required. For example, to define the ? SET query command as the PF3 key, type:

```
PF3 FOCUS ? SET
```

Note:

- You may also specify the WINDOW SET command from the Command Window to change a key setting for the session.
- PF key assignments revert back to the default settings when you end a FOCUS session. To retain customized key settings for each session, define them with the WINDOW SET command in your PROFILE FOCEXEC.
- If you exit the Terminal Operator Environment to return to the FOCUS command level, PF key assignments are retained in the Help Window when you reenter the optional environment.
- The FOCUS HELP facility is available from this environment; issue the FOCUS HELP command from the Command Window.

Table Window

The Table Window displays the results of the most recent TABLE request. This enables you to view the report again without resubmitting the request. Unlike the RETYPE command, the most recent report is available even if other commands have been issued after the request.

The Table Window displays a TABLE report as soon as you have terminated the report in HotScreen. The Table Window holds up to the first 10 pages of report data (200 lines), up to a width of 130 characters.

Note: The Table Window does not record TABLEF reports, offline reports, or reports issued while the FOCUS SET SCREEN command is set to OFF.

Error Window

When a FOCUS error occurs, the Error Window appears in the middle of the screen and displays an error message. The Error Window always has a bright border, even when it is not the active window. It remains on the screen until the error is corrected.

When you issue a command from the Command Window, it is copied to the Output Window and the History Window. The command is processed and FOCUS checks for errors. If an error is detected, the Error Window appears and the cursor positions itself in the Command Window. If part of the command is correct and has been accepted by FOCUS, that part is protected.

At this point, you have two choices:

- Correct the error identified by the cursor, add any new lines if you wish, and press *Enter* to resubmit the command.
- Terminate the command without executing it. Enter the QUIT command at the current cursor position and delete any leftover characters.

You may also control the length of the error message that appears in the Error Window. Use the WINDOW SET ERRORS command to specify long form or short form.

Displaying Fields and Field Formats

The Output Window displays a list of fields and accompanying aliases and formats when you issue the ?FF or ?F query command outside of a request.

Before you enter your request from the Command Window, issue the ?FF query or ?F command:

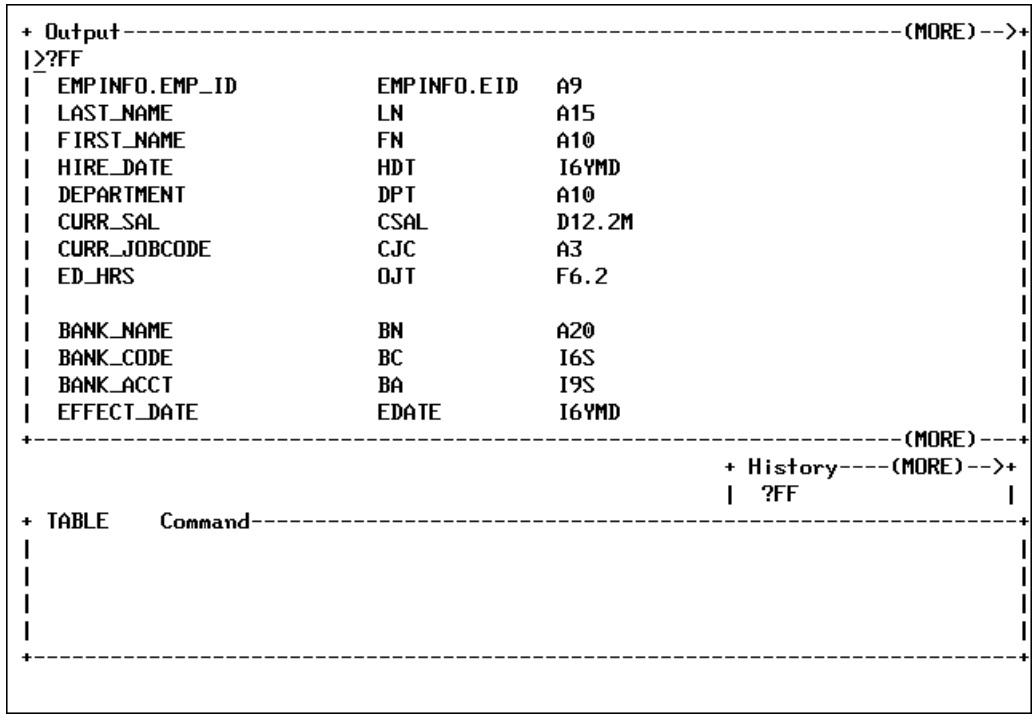
```
?FF
```

```
?F
```

The Output Window displays the fields. Note the field you wish to use in the request and then press the *PF12* key to return to the Command Window. Type the request with the field you selected and press *Enter*.

To verify that the field has been added to the request, make the History Window the active window. Press *PF2* to zoom in; the request with the added field appears. Press *PF2* to zoom out. Then make the Command Window the active window again so that you can continue creating your request.

In the following screen, the ?FF EMPLOYEE command was entered in the Command window. When the *Enter* key is pressed, the Output Window displays the list of fields.



Window Commands

In the Terminal Operator Environment, several WINDOW commands are available to control features, window behavior, and screen design.

The syntax for a WINDOW command requires the keyword WINDOW. For some commands, the name of the window is optional. If you do not specify a window in these cases, the active window is assumed by default. You can also use unique truncations for every word in the command.

WINDOW commands are issued from the Command Window, although some have associated PF keys. (You may assign WINDOW commands to PF keys as described in *Help Window: Revising PF Key Settings* on page 94.) Commands that you use often may be stored in your PROFILE FOCEXEC.

- The ACTIVE and NEXT commands activate a window.
- The CLEAR command clears the contents of a window.

- The SET CONTINUE, SET AUTOSCROLL, and SET IMMEDTYPE commands control the behavior of the Output Window.
- The CLOSE, OPEN, MOVE, SET ERRORS, SET, and SIZE commands customize your screen.
- The HELP command displays the Help Window.
- The ZOOM command enlarges a window.
- The RECALL command recalls issued commands.
- The ROUTE command routes the contents of a window to a data set.
- The SCROLL command controls the display of the window contents.

Note: In the syntax that follows, the term windowname is used to denote the Command Window, the Output Window, the History Window, the Help Window, or the Error Window.

Commands for Activating a Window

There are two commands that activate a window: the ACTIVE command and the NEXT command.

Syntax: How to Activate a Window

```
WINDOW ACTIVE windowname
```

When you issue the ACTIVE command from the Command Window, the specified window becomes highlighted and the Command Window becomes deactivated. If the specified window is not displayed on the screen, it appears and overlays existing windows.

For example, to activate the History Window, you would enter:

```
WINDOW ACTIVE HISTORY
```

Syntax: How to Activate the Next Window in the Screen Sequence

```
WINDOW NEXT
```

Pressing PF12 is equivalent to issuing the NEXT command.

Clearing a Window

The CLEAR command erases the contents of a window.

Syntax: How to Erase the Contents of a Window

```
WINDOW CLEAR [windowname]
```

For example, to clear the Output Window, enter:

```
WINDOW CLEAR OUTPUT
```

The contents of the Output Window (including data that is not visible) are erased; data in the History Window is not affected.

Controlling the Output Window

The following commands control the contents of the Output Window: SET AUTOSCROLL, SET CONTINUE, and SET IMMEDTYPE.

Syntax: How to Automatically Scroll the Contents of the Output Window

```
WINDOW SET AUTOSCROLL {ON|OFF}
WINDOW SET AUTOSCROLL OFF
```

where:

ON

Automatically scrolls the Output Window down. If the new set of output will not fit in the remaining window space, the display begins at the top of the Output Window. This value is the default.

OFF

Begins displaying output on the next available line of the Output Window. The window is scrolled only when it is filled.

For example, to prevent the Output Window from automatically scrolling, enter:

```
WINDOW SET AUTOSCROLL OFF
```

Syntax: How to Control Data Transmission

```
WINDOW SET CONTINUE {ON|}
```

where:

ON

Waits until the executing procedure has finished and transmits data to the Output Window until the next input from the terminal is received (for example, until you press a key), or until there is no more data.

OFF

Pauses when transmitting a stream of data to the Output Window each time the window is filled. To continue the data transmission, press *Enter*. This value is the default.

For example, if you plan to execute a procedure that generates several screens of output and you do not want FOCUS to pause when the Output Window becomes full, enter:

```
WINDOW SET CONTINUE ON
```

In this case, you do not see any data in the Output Window until the entire procedure is completed and FOCUS prompts you for input.

Syntax: How to Control Buffering of Line Mode Output

```
WINDOW SET IMMEDTYPE {ON|OFF}
```

where:

ON

Sends all line mode output, such as -TYPE to the Output Window as it is executed, line by line.

OFF

Buffers all line mode output. The output appears in the Output Window as a new full screen. This value is the default.

Customizing Your Screen

The Terminal Operator Environment screen is built with solid borders to enhance the display on terminals that support this feature. If your terminal does not support solid borders, set the parameter as follows

```
SET SBORDER=OFF
```

before entering the Terminal Operator Environment.

There are several window commands that control the layout of windows on the screen and that define the PF keys. You can use these commands to customize the Terminal Operator Environment.

- The CLOSE command removes a window from the screen.
- The OPEN command displays a closed window in its normal screen location.
- The MOVE command moves a window to a new screen location.

- The SET ERRORS command controls the length of the error message that displays in the Error Window.
- The SET command is used to redefine key settings.
- The SIZE command changes the height or width of a window.

Syntax: How to Remove a Window From the Screen

```
WINDOW CLOSE [windowname]
```

For example, if you do not want to see the History Window, enter:

```
WINDOW CLOSE HISTORY
```

Your commands are recorded in the History Window even though it is not displayed.

Syntax: How to Display a Closed Window in its Normal Screen Location

```
WINDOW OPEN [windowname]
```

The window overlays existing windows. This command does not activate the window. This command also redisplays an opened window that is hidden behind other windows.

For example, to open the closed History Window, enter:

```
WINDOW OPEN HISTORY
```

Syntax: How to Move a Window to a New Screen Location

```
WINDOW MOVE [windowname] location {n|*}
```

where:

location

Is one of the following:

ROW moves the top border of the window to row *n*, an absolute position.

COLUMN moves the left border of the window to column *n*, an absolute position.

LEFT moves the window to the left. If *n* is specified, the window moves *n* columns to the left. If asterisk (*) is specified, the left border of the window moves to the left edge of the screen.

RIGHT same as LEFT, but to the right.

`UP` moves the window up. If *n* is specified, the window moves up *n* columns. If asterisk (*) is specified, the top border of the window moves to the top edge of the screen.

`DOWN` same as UP, but the window moves down and the bottom border becomes the bottom edge of the screen.

n

Is any positive number.

*

Used with LEFT, RIGHT, UP, and DOWN. Moves the window to the edge of the screen.

For example,

To move the History Window up to Row 10, enter:

```
WINDOW MOVE HISTORY ROW 10
```

To move the Table Window up 12 rows, enter:

```
WINDOW MOVE TABLE UP 12
```

Note:

- If the specified screen location causes any part of the window to extend past the physical screen, the window is moved only to the edge of the screen.
- Windows return to their default positions when the FOCUS session is terminated unless the positions are specified in your PROFILE FOCEXEC.

You may also use the PF9 key to position windows. The MOVE CURSOR command is only available as a PF key setting and cannot be issued as a command from the Command Window. The syntax is:

```
MOVE [windowname] CURSOR
```

To move a window using PF9, position the cursor at the new location and press PF9. The top left corner of the window is moved to the current cursor position. If the window disappears from the screen, press PF12 to activate it again.

Syntax: How to Control the Length of an Error Message in the Error Window

```
WINDOW SET ERRORS {SHORT|LONG}
```

where:

`SHORT`

Displays the short form: the error number and description.

`LONG`

Displays the long form: the error number, description, and an explanation. This value is the default.

For example, to display error messages without explanations, enter:

```
WINDOW SET ERRORS SHORT
```

Syntax: How to Redefine Key Settings

```
WINDOW SET key command
```

where:

key

Is any PF key.

command

Is any WINDOW or FOCUS command.

If you assign a WINDOW command to a PF key, do not include the WINDOW keyword. For example, to set PF14 to the WINDOW CLOSE command, enter:

```
WINDOW SET PF14 CLOSE
```

If you assign a FOCUS command to a PF key, the keyword FOCUS is required. For example, to assign the ? SET query command to the PF4 key, enter:

```
WINDOW SET PF4 FOCUS ? SET
```

Note:

- You can edit the Help Window and immediately change the key settings (see *Help Window: Revising PF Key Settings* on page 94).

- Key settings change back to their default settings when you end the FOCUS session unless they are defined in the PROFILE FOCEXEC.

Syntax: How to Change the Height or Width of a Window

```
WINDOW SIZE [windowname] {WIDTH|HEIGHT} {n|*} {MORE|LESS}
```

where:

`WIDTH`

Changes the width of the window.

If you alter the width, the right window side is increased or decreased accordingly.

`HEIGHT`

Changes the height of the window.

If you change the height, the bottom of the window is increased or decreased accordingly.

n

Is any positive number. Indicates an absolute size or a size relative to the existing size if MORE or LESS is specified.

`*`

Extends the width or height of the window to that of the physical screen.

`MORE`

Increases the window size by n columns or rows. Not used with asterisk (*).

`LESS`

Decreases the window size by n columns or rows. Not used with asterisk (*).

For example, to increase the height of the History Window by five rows, enter:

```
WINDOW SIZE HISTORY HEIGHT 5 MORE
```

The MORE option indicates relative sizing; omit it for an absolute size. For example, to make the History Window five rows high, enter:

```
WINDOW SIZE HISTORY HEIGHT 5
```

Note: If the new window size causes any part of the window to extend beyond the physical screen, the window is sized only to the edge of the screen.

Displaying the Help Window

The HELP command controls the display of the Help Window. It opens and activates a closed Help Window. Issue this command again to deactivate and close it.

Syntax: How to Display the Help Window

```
WINDOW HELP
```

Pressing the *PF1* key is equivalent to issuing the HELP command. Press the key once to open the Help Window; press the key again to close it.

Enlarging a Window

The ZOOM command enlarges a window up to the full size of the screen. It also shrinks an enlarged window to its normal size. The specified window becomes active as a result.

Syntax: How to Enlarge a Window

```
WINDOW ZOOM [windowname]
```

Pressing the *PF2* key is equivalent to issuing the ZOOM command. Move the cursor and press *Enter* to activate the window. Then, press *PF2*.

Note: A blank window results from enlarging a closed Help Window. Display the window first and then issue the ZOOM command.

Recalling Commands

The RECALL command is only available as a PF key setting. Although the RECALL command is the current default for PF6, you may assign

```
RECALL
```

to any PF key. There are two ways to use the PF6 key:

- Press the *PF6* key to recall the most recent command. If you continue to press the *PF6* key, previously entered commands appear according to the order in which they were entered.
- Position the cursor in an active History Window next to the command and press *Enter*.

This recalls a command from the History Window to the Command Window. You can edit the command once it is recalled to the Command Window and submit it instead of typing it again. The command remains in the History Window for future use.

Note: A FOCUS request with several subcommands (like a TABLE request) is treated as one command; therefore, the entire request appears when you press *PF6*.

Routing Window Contents

The ROUTE command transfers window contents to an allocated file or data set while it continues to display the contents in the window. To stop the routing, issue the ROUTE command again with the OFF option. With this command, you can create a session monitor record or log by routing the History Window or the Output Window to a file.

Syntax: How to Route Window Contents

```
WINDOW ROUTE [windowname] {TO ddname|OFF}
```

where:

ddname

Is any valid ddname.

OFF

Will stop routing data to the ddname.

For example, to route History Window contents to a file allocated to ddname SESSION, enter:

```
WINDOW ROUTE HISTORY TO SESSION
```

Note: You must issue an ALLOCATE command before you issue the ROUTE command; space allocation is not set dynamically. The ddname should be allocated to a sequential data set with LRECL 132 and RECFM F.

Scrolling Window Contents

The SCROLL command moves the window contents when data extends beyond the window border and the MORE message or right and left indicators (< or >) appear. You can scroll a window in any direction.

Syntax: How to Scroll Window Contents

```
WINDOW SCROLL [windowname] direction
```

where:

direction

Is one of the following:

FORWARD scrolls the window down; also available as the PF8 key.

`BACKWARD` scrolls the window up; also available as the PF7 key.

`TOP` scrolls the window to the top line; also available as the PF4 key.

`BOTTOM` scrolls the window to the bottom line; also available as the PF5 key.

`LEFT` scrolls the window left; also available as the PF10 key.

`RIGHT` scrolls the window right; also available as the PF11 key.

Note:

- Using the PF key instead of entering the SCROLL command from the Command Window is recommended, as the automatic autoscroll feature might override your explicit SCROLL command.
- You may also scroll the Output Window forward with the PA2 or CLEAR key while you are working in another active window.

Chapter 5

UNIX and Linux Guide to Operations

FOCUS is currently available on many UNIX platforms. The *FOCUS for Distributed Systems UNIX/Linux Installation Guide* lists the platforms on which FOCUS is available and describes the installation procedure. While it is possible to run FOCUS under any of the available shell environments, it is tested and certified only under the Bourne (sh) shell, as supplied by the original hardware vendor.

In this chapter:

- How FOCUS Interacts With UNIX
- Overview of a FOCUS Session
- Defining the FOCUS Operating Environment
- Defining and Allocating FOCUS Files
- FOCUS in the UNIX and Linux Environments
- Sending Email From a Procedure
- Configuring, Starting, and Stopping the FOCUS Database Server

How FOCUS Interacts With UNIX

This section describes how FOCUS interacts with a UNIX environment.

Accessing FOCUS

In order to execute a UNIX command, you must either specify the full path name or add the appropriate directory to your UNIX PATH variable before you issue the command.

FOCUS is installed in several subdirectories of a parent directory. The various FOCUS shell scripts reside in the tools subdirectory. This subdirectory is the one you should specify in the full path name or in your UNIX PATH variable.

The following example shows how to add the foc/bin subdirectory to your PATH, assuming that FOCUS resides in /usr/ibi:

```
PATH=$PATH:/usr/ibi/srv82/foc/bin
export PATH
```

UNIX Directory Permissions

FOCUS respects UNIX file permissions for reading and writing files. In carrying out requests, FOCUS relies on a number of temporary work files. In order to create and use these work files, write permission for the directory from which FOCUS was started is required.

Terminal Support

FOCUS uses the information in the UNIX terminfo files to determine how to interact with full-screen devices. In order for FOCUS to interact correctly with your hardware, the name of the corresponding terminfo file must be assigned to the the UNIX variable TERM and the variable must be exported before you start FOCUS. An incorrect TERM value or an inaccurate terminfo file may lead to problems with your screen display or keyboard. If you encounter problems, see *FOCUS in the UNIX and Linux Environments* on page 125, for information on how to specify device attributes.

The FOCUS Operating Environment

There are several ways in which you can define the operating environment for FOCUS. You can:

- Set and export UNIX environmental variables before you start FOCUS.
- Specify optional parameters when you issue the focus command to start FOCUS.

For more information, see *Defining the FOCUS Operating Environment* on page 112.

Overview of a FOCUS Session

This section presents an overview of a FOCUS session.

The focus Shell Script

The focus command invokes a shell script, which performs the following functions and then starts the binary program:

1. Assigns default values to UNIX variables that create the desired environment for the executable code. For information on what these variables control and how to assign a value other than the default, see *Defining the FOCUS Operating Environment* on page 112.
2. Checks whether the user has write access for the current directory. If the user does not have appropriate access, the script issues an error message and does not start FOCUS.

Beginning and Ending a FOCUS Session

To begin a FOCUS session, issue the following command at the UNIX prompt:

```
focus
```

This command invokes the focus shell script. You can specify optional parameters that determine FOCUS behavior immediately following the focus command. For a list of options, see *Defining the Environment Using Command-Line Parameters* on page 112.

FOCUS displays one of the following.

- The FOCUS prompt (>>). To use FOCUS, issue the desired commands at the prompt. To end the session, issue the following command at the prompt:

  ```
  FIN
  ```

- A site-specific application. To use the application or end the session, follow the directions provided by the application.

Profile Processing

FOCUS supports the following levels of profiles to provide flexibility in designing and running applications.

- A **global profile** is a startup file that is automatically created during the FOCUS installation and configuration. It contains the default environment settings that are required for the proper FOCUS operation.
- A **group profile** is a file used by FOCUS to specify settings that apply to a FOCUS environment, but only for a user of a specific security group.
- A **user profile** is available, based on a user ID. A user profile is a file used by FOCUS to specify settings that apply to a FOCUS environment, but only for a specific user ID.

When you start FOCUS, it searches in the current directory for a FOCUS procedure called profile.fex. If this file exists, FOCUS executes it. If there is no profile.fex in the current directory, FOCUS searches the directories specified in the FOCUS search path for profile$.fex, executing the first one it finds. If FOCUS finds neither of these profiles, it displays the prompt and awaits further instructions.

Interrupting a FOCUS Session

To stop a running program during a FOCUS session, press the UNIX Interrupt key, which sends an operating system signal to interrupt the session and close the files. On most systems, the Interrupt key is either Ctrl+C or Delete. If you do not know which key to use, issue the following UNIX command:

```
stty -a
```

The attribute *intr* identifies the Interrupt key.

Defining the FOCUS Operating Environment

There are several ways in which you can define the operating environment for FOCUS. You can:

- Set and export UNIX environmental variables before you start FOCUS. For more information, see *Defining the Environment Using UNIX Variables* on page 112.
- Specify optional parameters when you issue the *focus* command to start FOCUS. For more information, see *Defining the Environment Using Command-Line Parameters* on page 112.

Defining the Environment Using UNIX Variables

UNIX environmental variables enable you to define the operating environment for FOCUS. The focus shell script assigns default values to these variables. If you wish to assign values other than the default, you can declare the desired variables and export them before you start FOCUS. In addition, you can assign values for some of the variables on the focus command line.

To declare a variable and assign a value before you start FOCUS, issue the following commands:

```
variable=value
export variable
```

where:

`variable`

Is the name of the variable you wish to declare.

`value`

Is the value you wish to assign.

`export`

Is a UNIX command that makes the variable available to all subsequent processes.

If the values you wish to assign are permanent, consider including the declarations in your UNIX profiles, the MI file, or in a .*rc file in your home directory.

Environment variables can be set and then read while in the FOCUS environment using the FGETENV function.

Defining the Environment Using Command-Line Parameters

You can define the FOCUS operating environment using command-line parameters. The syntax is:

```
focus [option ... option]
```

The following list summarizes the available options:

Option	Description
`-f script`	Starts FOCUS, with input from file *script*.t3i, and output to file *script*.t3o.
`-x "command"`	Starts FOCUS with a single command as input. Output goes to the terminal.
`-sinkstart`	Starts the FOCUS Database Server (sink machine).
`-sinkstop`	Stops the FOCUS Database Server (sink machine).
`-savediag`	Is used in conjunction with -traceon. Provides an option to gather and package trace files and other diagnostic information for Customer Support Services.
`-traceon`	Turns on tracing.
`-traceoff`	Turns off tracing.

You can specify more than one option. However, if you do so, you must precede each option with a dash (-). You cannot combine options according to the UNIX convention of using a single dash followed by a list of options.

Defining and Allocating FOCUS Files

You will come across a variety of files when you use FOCUS. These files can be application files that you create, extract files that FOCUS creates as requested by your application, or work files that FOCUS creates as required by your application.

This topic describes FOCUS file allocation requirements and describes how to assign a logical name, or DDNAME, to a file or device for operating systems that support this feature.

Allocating FOCUS Files

FOCUS files fall into three categories: application files, extract files, and work files. This topic contains information about files and ways to define them in the UNIX and Linux environments.

Some files are automatically allocated by FOCUS, but there are times when you must explicitly define a file and its location. A FILEDEF command generates platform independent file paths for all portable platforms by creating FILEDEF syntax with application names.

Reference: FOCUS Files Under UNIX and Linux

The following tables describe the FOCUS files that you will use under UNIX and Linux. FOCUS uses file extensions to distinguish different file types.

Application Files

Extension	Description
`.mas`	Master File.
`.acx`	Access File.
`.fex`	Procedure (FOCEXEC).
`.foc`	FOCUS and XFOCUS data sources and external index.
`.dat`	Sequential data source.
`.sty`	FOCUS StyleSheet file.
`.err`	Error messages file or help text.
`.prf`	Profile.

`.htm`
`.html`
`.jpeg`
`.gif`
`.css`
`.js`
`.class`
`.jar`
	Files used for HTML files and styling for HTML files.

Extract Files

File	Description
HOLD	Contains data saved using the HOLD command.
SAVB	Contains data saved using the SAVB command.
SAVE	Contains data saved using the SAVE command.
.FTM	Contains data saved using the HOLD, SAVB, or SAVE command.

Note: Dialogue Manager output files must be allocated using the FILEDEF command in system or user profiles.

Work Files

File	Description
FOCSORT	Used during sorting.
FOCPOST	Sequential output file saved using the POST command. The PICKUP command reads it back in.
FOCSML	Used by Financial Modeling Language (FML).
OFFLINE	Used when the SET PRINT parameter is OFFLINE.
SYSIN	Directs input.
SYSPRINT	Directs output to the screen.

Dynamically Defining a File Under UNIX and Linux

You do not have to explicitly define most of your application files prior to referring to them. FOCUS dynamically allocates certain application files. Additionally, FOCUS dynamically defines all extract files and temporary work files to the operating system during a session.

FOCUS defines the following extract, output, and work files:

- HOLD, SAVB, and SAVE files.

- FOCUS and XFOCUS data sources.
- FOCSORT files.
- SYSIN to input from keyboard. It is assigned to TERM. This is equivalent to the command FILEDEF SYSIN TERM.
- SYSPRINT to output to be displayed on the screen. It is assigned to TERM. This is equivalent to the command FILEDEF SYSPRINT TERM.
- OFFLINE to output sent to the printer. It is assigned to PRINTER. This is equivalent to the command FILEDEF OFFLINE PRINTER.

Assigning a Logical Name With the FILEDEF Command

For a file managed by the operating system, such as an ISAM or comma-delimited data file, the physical file name is the actual name of a file as it appears to the operating system. A logical name (or ddname) is a shorthand name that points to the physical file name. Logical names simplify code by allowing short names to be used in place of the longer physical file name.

The FILEDEF command assigns a logical name to a physical file name and specifies file attributes. You can explicitly define a file and its location to FOCUS using the FILEDEF command. This generates platform independent file paths for all portable platforms by creating FILEDEF syntax with application names. An allocation can be issued in a procedure and lasts for a single request.

It is recommended that instead of including an allocation in each procedure, you include all FILEDEF commands in a single file that you call with the -INCLUDE command at the beginning of each procedure. This enables you to make changes to your FILEDEF commands globally instead of changing the allocation information in each procedure.

The FILEDEF command is typically used in the following ways in operating systems that support this command:

- **Identifying data sources.** You must create FILEDEF assignments (or USE directories, in the case of FOCUS data sources) for all data sources you wish to use.
- **Redirecting output sent to the printer.** The OFFLINE ddname is used for output sent to the printer. By reassigning this ddname, you can redirect printer output to a file.

You can also use the Universal Naming Convention (UNC) to assign logical names to files that are located on a server. In order to take advantage of the UNC you must first attach to the server you want to use. For information on attaching to a server or mapping network drives, consult your network administrator.

Syntax: How to Assign a Logical Name With a FILEDEF Command

```
FILEDEF ddname DISK appname[/appnamea...]/filename [(APPEND] [LRECL n]
[RECFM F]
```

where:

ddname

Is the logical name for the file. The *ddname* can be from one to eight characters. When used to associate a data source with a Master File, the *ddname* must match the name of the Master File.

DISK

Associates the specified *ddname* with a file.

appname[*/appnamea...*]

Is the name of the application under APPROOT, or a nested application name, that contains the physical file.

filename

Is the physical name of the file under the appname.

APPEND

Appends records to the end of the file. Without this option, the file is overwritten.

LRECL *n*

Specifies the record length. *n* is an integer.

You must specify an LRECL value if the file is a SAVB file or fixed-format transaction file used in data maintenance (FIXFORM files) that contains binary values.

RECFM F

Specifies fixed length records.

You must specify an RECFM value if the file is a SAVB file or fixed-format transaction file used in data maintenance (FIXFORM files) that contains binary values.

FOCUS data sources (files with the .foc extension) that do not conform to the default naming convention are identified with the USE command, not FILEDEF. For information on the USE command, see the *Describing Data* manual.

Example: Redirecting Output

To send output to the LPT1 port (provided that the machine is configured properly):

```
FILEDEF OFFLINE DISK LPT1
```

Example: **Setting a Search Path**

To search all directories on the search path for the NEW_EMPS.DAT file, set your APP PATH. For example:

```
APP PATH app1 app2 app3 ... appn
```

Example: **Appending a Report Extract to Other Content**

To append a report extract from the LIBRARY data source to the current content of the file LIB03.FTM:

```
FILEDEF SAVE DISK C:\LIBRARY\LIB03.FTM (APPEND
```

Example: **Reading Standard Text Editor Files With LRECL**

You can specify an LRECL equal to or greater than the implied LRECL for the request. For example, if the length of the longest line in the text file is seven characters, issue this command:

```
FILEDEF BIGLINE DISK BIGLINE.FTM (LRECL 7
```

Displaying Current ddnames Assigned With FILEDEF

The ? FILEDEF command displays the ddnames assigned for various files, input and output.

Syntax: **How to Display Current ddnames**

```
? FILEDEF
```

Example: **Displaying Current ddnames**

Issuing the command

```
? FILEDEF
```

produces information similar to the following:

```
Lname   Device   Lrecl   Recfm   Append   Expl   Filename
============================================================
HOLD2   DISK       0     V       N        Y      C:\VM\SMALL\HOLD2.FTM
```

Clearing Allocations

You can clear allocations by using FOCUS syntax.

Syntax: How to Clear a Logical Name With Syntax

```
FILEDEF ddname CLEAR
```

where:

ddname

Is the logical name. It may contain one to eight alphanumeric characters.

CLEAR

Clears the specified ddname.

Application Files Under UNIX and Linux

This topic describes how FOCUS references and searches for application files.

When you do not change the default file type, you can prepare a report without issuing a FILEDEF command or other communication. When you change the default file type of a FOCUS data source, or the data source does not reside in a standard search path, you must issue the USE command. The USE command is discussed in the *Describing Data* manual.

Master Files Under UNIX and Linux

A Master File contains metadata about a data source, and is identified with a file name and extension, for example, *filename*.mas. It consists of attributes that describe a data source. A Master File and the data source that it describes usually have the same name.

The description of a data source and all files it cross-references must be available whenever you refer to the data source.

For more information about how to generate Master Files, see the *Describing Data* manual.

Reference: Naming a Master File Under UNIX and Linux

The following rules apply:

- The file name can be up to 64 characters long. The first character should be a letter, and the remaining characters can be any combination of letters, numbers, and underscores. Other special characters may be used, but could cause problems and are therefore not recommended or supported.
- The file extension must be .mas for a Master File, since FOCUS searches for a Master File with that extension by default.

Access Files Under UNIX and Linux

For some data sources, an Access File supplements a Master File. An Access File includes additional information that completes the description of the data source. For example, it includes the full data source name and location. You need one Master File, and for some data sources one Access File, to describe a data source.

FOCUS searches for an Access File on the path using APP PATH logic.

Reference: Naming an Access File Under UNIX and Linux

The following rules apply:

- The file name can be up to 64 characters long. The first character should be a letter, and the remaining characters can be any combination of letters, numbers, and underscores. Other special characters may be used, but could cause problems and are therefore not recommended or supported.
- The file extension must be .acx since FOCUS searches for a FOCUS data source with that extension by default.

Procedures Under UNIX and Linux

A procedure contains your report requests and cannot have control characters. A procedure is identified with a file name and extension, for example, *filename*.fex.

Procedures can be simultaneously accessed and executed by all users. The device and directory must precede the procedure file name and extension for FOCUS to locate the procedure. If you specify a disk and a directory, you must include both the file name and the file extension (.fex) when calling the procedure.

FOCUS searches for a procedure on the path, using APP PATH logic.

Reference: Naming a Procedure Under UNIX and Linux

The following rules apply:

- File names can be up to 64 characters long. The first character should be a letter, and the remaining characters can be any combination of letters, numbers, and underscores. Other special characters may be used, but could cause problems and are not recommended or supported.
- The file extension must be .fex, since FOCUS searches for a procedure with that extension by default.

FOCUS and XFOCUS Data Sources Under UNIX and Linux

FOCUS and XFOCUS data sources contain data written in FOCUS format. All FOCUS data sources have a record length of 4096 and a fixed length record format. The maximum size of a FOCUS data source is 2G. XFOCUS data source have a record length of 16K and a fixed length record format. XFOCUS data sources can be a maximum size of 16G.

A FOCUS or XFOCUS data source is identified with a file name and extension, for example, *filename*.foc for both types of data sources. The name of a FOCUS data source usually matches the name of the corresponding Master File. For example, if the Master File is ledger.mas, then the FOCUS or XFOCUS data source is ledger.foc. You can override this default with the USE command, described in the *Describing Data* manual.

Reference: **Naming a FOCUS Data Source Under UNIX and Linux**

The following rules apply:

- The file name can be up to 64 characters long. The first character should be a letter, and the remaining characters can be any combination of letters, numbers, and underscores. Other special characters may be used, but could cause problems and are therefore not recommended or supported.
- The file extension must be .foc since FOCUS searches for a FOCUS data source with that extension by default.

External Indexes for FOCUS Data Sources Under UNIX and Linux

An external index is a FOCUS data source that contains index, field, and segment information for one or more specified FOCUS data sources. The external index is independent of its associated FOCUS data source and is used to improve retrieval performance. In UNIX and Linux, the external index is automatically allocated as a permanent file when it is created using REBUILD.

Sequential Data Sources Under UNIX and Linux

FOCUS recognizes sequential data sources formatted in the following ways:

- Fixed-format, in which each field occupies a predefined position in the record.
- Free-format, also known as comma-delimited, in which fields can occupy any position in a record.
- Delimited, in which fields are separated by a wide variety of delimiter characters.

For details, see the *Describing Data* manual.

FOCUS StyleSheets Under UNIX and Linux

FOCUS StyleSheets enable you to style and produce reports that highlight key information. With StyleSheets, you specify various stylistic characteristics of a report and style report components individually. You can also use StyleSheets to define a hyperlink from any reporting object.

You can create StyleSheets externally or within a report request. For details, see the *Creating Reports* manual.

FOCUS searches for a FOCUS StyleSheet on the path, using APP PATH logic.

Reference: **Naming a StyleSheet Under UNIX and Linux**

The following rules apply:

- The first character should be a letter, and the remaining characters can be any combination of letters, numbers, and underscores. Other special characters may be used, but could cause problems and are not recommended or supported.
- The file extension must be .sty since FOCUS searches for a StyleSheet with that extension by default.

Extract Files Under UNIX and Linux

FOCUS creates all extract files in a temporary directory, unless you use an APP HOLD command, an APP HOLDDATA command, or FILEDEF the file to a permanent file in a permanent directory.

HOLD Files Under UNIX and Linux

A HOLD extract file is a sequential data source that contains the results of a report request. It may have a corresponding HOLD Master File. The extension for a HOLD file is .ftm unless you specify the FORMAT option. If a Master File is created, it has the same name as the extract file, with the extension .mas.

Syntax: **How to Create a HOLD File Under UNIX and Linux**

```
ON TABLE HOLD [AS [app/]filename]
```

where:

app
Is the name of the application directory in which to create the file. Both the data file and the Master File are created in this application directory.

filename

Is a name for the HOLD file. If you do not specify a file name, HOLD is used as the default name. Since each subsequent HOLD command overwrites the previous HOLD file, it is useful practice to code a distinct file name in each request to direct the extracted data to a separate file, preventing it from being overwritten.

For the complete syntax with all options, see the *Creating Reports* manual.

SAVB Files Under UNIX and Linux

A SAVB file is an extract file containing the results of a report request in internal format. That is, all numeric fields are stored in binary, and all character fields are padded with spaces to a multiple of four bytes. The file cannot be printed.

Syntax: How to Create a SAVB File Under UNIX and Linux

```
ON TABLE SAVB [AS [app/]filename]
```

where:

app
Is the name of the application directory in which to create the file.

filename

Is the name of the file. The default is SAVB. The default extension is .ftm.

SAVE Files Under UNIX and Linux

A SAVE file is an extract file that saves the data of a report, but does not save headings, or subtotals, or create a Master File. It is a simple sequential character data file and can be used by other programs, or merged into another data file using a data maintenance request. The default format is simple character, although you can specify formats compatible with many other software products. FOCUS saves all columns in the report in printable, character format with no spaces between columns.

A SAVE file contains the external character format equivalent to SAVB. The command format and allocations are the same as SAVB. However, the numbers are printable American Standard Code for Information Interchange (ASCII) characters and no padding takes place.

Syntax: How to Create a SAVE File Under UNIX

```
ON TABLE SAVE [AS [app/]filename]
```

where:

app
Is the name of the application directory in which to create the file.

filename

Is the name of the file. The default is SAVE. The default extension is .ftm.

HOLDMAST Files Under UNIX and Linux

A HOLDMAST file is a temporary Master File. Master Files for HOLD files will be created in a temporary directory unless the HOLD command contains an application name (app/filename) or an APP HOLD or APP HOLDMETA command is issued, which specifies its location.

If the HOLD file was created under the default name HOLD, the FILEDEF command can be used in conjunction with the APP HOLDMETA syntax to designate a logical name to the HOLD file and respective Master File.

Syntax: How to Specify the Location for a HOLD Synonym Under UNIX

```
APP HOLDMETA app_name
```

where:

app_name

Is a valid application directory.

Example: Allocating a Logical Name for a HOLD File and Location for the Master File Under UNIX and Linux

The following example sets a logical name for the HOLD file with FILEDEF and specifies a location for the Master File with APP HOLDMETA:

```
FILEDEF SALES DISK app1/sales.ftm
APP HOLDMETA app1
TABLE FILE GGSALES
SUM DOLLARS
BY CATEGORY
ON TABLE HOLD AS SALES
END
```

The name of the Master File is derived from the ON TABLE HOLD AS *filename* name, which specifies *sales* as the logical name for the physical data source (sales.ftm). The Master File is given the same file name with the extension .mas (sales.mas). Both of these files are placed in application directory app1 as permanent files.

Note: Other methods for creating both the HOLD file and Master File in an application directory include the following:

- Specifying the application directory name in the HOLD command (HOLD AS app1/sales).
- Issuing the APP HOLD command (APP HOLD app1).

To report on the HOLD file, you need to issue a FILEDEF command (unless the Master File has a DATASET attribute that points to the file). For example:

```
FILEDEF SALES DISK app1/sales.ftm
TABLE FILE SALES
PRINT *
END
```

Work Files Under UNIX and Linux

FOCUS creates work files as required by your application. FOCUS creates all extract files in a temporary directory unless you issue an APP HOLD command or FILEDEF the file to a permanent directory.

The available work files are:

- **FOCSORT**, allocated as FOCSORT.FOC, may be required for sorting with the TABLE, TABLEF, and MATCH commands.
- **FOCPOST** files hold sequential output, which is saved by the POST command and read by the PICKUP command.
- **FOCSML** files are used by the Financial Modeling Language (FML) facility.
- **OFFLINE** sends output to a file when the PRINT parameter is set to OFFLINE. Allocate the file to ddname OFFLINE using a FILEDEF command after the offline has been closed with the OFFLINE CLOSE command.

Example: **Sending Output With the OFFLINE Command Under UNIX and Linux**

The following command sends further report output to the file EMPL OUTPUT A with a fixed length record format and a record length of 80 bytes:

```
OFFLINE
FILEDEF OFFLINE DISK TMP/EMPL.OUT (RECFM F LRECL 80)
```

FOCUS in the UNIX and Linux Environments

This section describes how certain FOCUS facilities work in a UNIX or Linux environment.

Terminal Support

FOCUS uses the information in the terminfo files to determine how to interact with full-screen terminals. The UNIX environmental variable TERM holds the name of the file that describes the particular terminal being used.

In order for FOCUS to interact correctly with a particular terminal, the name of the corresponding terminfo file must be assigned to the the UNIX variable TERM and the variable must be exported before you start FOCUS. Since this is a UNIX standard, these steps are usually included in the UNIX profile and are executed automatically when a user logs in. To display the current value of TERM, issue the following UNIX command before you start FOCUS:

```
echo $TERM
```

To verify that the value has been exported, issue the following command:

```
export
```

To assign a value or change the current value of TERM and export it, issue the following commands:

```
TERM=value
export TERM
```

where:

value

Identifies the type of terminal you are using.

Note: If operating system messages, line noise, or other undesired characters overlay your screen during a FOCUS session, press Ctrl+L to refresh the screen. If this solution does not work, see your system administrator.

Remapping Keys

FOCUS supports terminal remapping, which enables you to translate the signals that are being transmitted from and to the terminal into values that are more useful to FOCUS users. The files fockeys.err and focescs.err control the interpretation of keyboard input and screen output, respectively. Directions for changing the input or output map appear at the beginning of the respective file.

Function Key Support

FOCUS uses the UNIX terminfo files to determine how to interact with full-screen terminals. In some cases, the terminfo files supplied with UNIX may not adequately support function keys. Also, a terminal may not have as many function keys as FOCUS allows. If your terminal lacks proper function key support, you can address the problem in one of the following ways:

- The system administrator can modify the UNIX terminfo files or the FOCUS fockeys.err file. Directions for changing fockeys.err appear at the beginning of the file.
- You can use the following method to specify the desired function key. Press the Esc key followed by two numeric keys. You do not need to press the Enter key. For example, to specify function key 1, enter:

  ```
  ESCAPE 0 1
  ```

  Note that function keys 1 through 9 require a leading zero.

International 8-Bit Character Support

FOCUS supports 8-bit characters. The terminal and keyboard you use determine how to get the desired character. Some keyboards provide the character directly. Others require a sequence of keys.

To determine whether your system is stripping the 8th bit before transmitting keyboard input to FOCUS, issue the following UNIX command before you start FOCUS:

```
stty -a
```

Look for the values cs8 and -istrip in the output. (istrip means strip the bit; -istrip means do not strip the bit.) To set the desired values, issue the following command:

```
stty cs8 -istrip
```

You can add this command to your UNIX profile.

The Concatenation Symbol

The FOCUS string concatenation symbol is the same as the UNIX pipe symbol. It can appear as a broken vertical bar (¦) or an unbroken vertical bar (|) and translates to ASCII 124 (HEX 7C, OCTAL 174).

Running FOCUS as a Background Process

The UNIX commands at, batch, and &, as well as redirection facilities, enable you to batch a process or run it in background. You can use these commands and facilities with FOCUS only if no interaction with the terminal is required. In addition, you should set the following FOCUS parameters to OFF: MORE, SCREEN, and PAUSE.

The following procedure, called tabbat.fex shows how to prepare a TABLE request to be run as a background process:

```
SET MORE=OFF, SCREEN=OFF, PAUSE=OFF
TABLE FILE CAR
PRINT COUNTRY
ON TABLE HOLD AS CARBATCH
END
FIN
```

The following command shows how to start FOCUS with the -p option, run the above request, redirect system messages to a file, and use & to send the process to background:

```
focus -x "ex tabbat.fex" &
```

While the above example uses redirection and &, similar results can be achieved with the other UNIX facilities mentioned previously. For information on these UNIX facilities, see your operating system documentation.

Sending Email From a Procedure

EDAMAIL is an internal procedure that allows you to send emails from a procedure with the content of a specific file included in the email body or as an attachment. This is useful for sending email alerts for events ranging from special error conditions to simple report completion. EDAMAIL requires a configured (and working) email SMTP Server node.

There are two forms of syntax for EDAMAIL:

- The original form uses positional parameters, and is limited to the original specification.
- The extended form uses non-positional, name-value pair parameters. The extended form supports additional SMTP email client features such as Reply To and Importance.

The extended form is assumed if the first parameter contains an equal sign (=) indicating that a name-value pair is being passed.

Most forms of email addresses are acceptable, including support@ibi.com, support1 <support@ibi.com>, and "support1" <support@ibi.com>. A semi-colon is used as the address separator for parameters that allow multiple email addresses. A comma is also allowed as a separator if the overall address string is enclosed in single quotation marks.

The actual use and display of the email headers created by EDAMAIL are a function of the SMTP Server in use (and any intervening SMTP hops), plus the user email client (such as Outlook, Gmail, Hotmail, and so on). They are not directly controlled by the sender email headers. Therefore, an email may not always exhibit the expected behavior in all email client environments. For example, older email clients may not recognize newer header types, and may ignore Reply To. Any functional issues should first be investigated by an experienced SMTP/Email administrator to determine if they are client-related.

Reference: **Configuring SMTP Server Parameters**

You must configure the SMTP Server node in the odin.cfg file. This configuration file is in the etc directory:

```
install_dir/ibi/srv82/foc/home/etc
```

The following declarations configure the SMTP Server node.

```
;SMTP mail server client
NODE = EMAIL
BEGIN
  SMTP_HOST = host
  PORT = port
  SENDER_EMAIL = email
  PROTOCOL = SMTP
  CLASS = SMTPCLIENT
END
```

where:

host
Is the domain name of the SMTP Server.

port
Is the port on which the SMTP Server is listening.

email
Is the sender email address for automatic emails.

Syntax: **How to Use EX EDAMAIL in the Positional Parameter Form**

The syntax is:

```
EX EDAMAIL to, from, subject, [flag], filetype, data
```

where:

to

Is the email recipient. Multiple addresses are allowed, using a semi-colon as the address separator.

from

Is the email sender.

subject

Is the email subject string. If the subject string contains a comma, the string must be enclosed in single or double quotation marks.

flag

If set to a or A, the file is sent as an attachment. Otherwise, it is included as the body of the email. All other values are ignored. The value is also ignored if the filetype parameter is blank.

filetype

Defines the data file type for an email message body that uses a file. Any application file type is valid, including MASTER, FOCEXEC, HTML, and TEXT. Leave the parameter empty to use the inline email message body feature.

data

Is the inline data stream for the email message body or the [app/]filename file containing the email message body. The inline data stream feature requires an empty filetype parameter. The EDAMAIL feature can also spread an inline email message body onto multiple lines using the -LINES {n|*} feature.

If an inline data stream message body is spread across multiple lines in the procedure, the resulting email is a single line of output. Multi-line message bodies are respected when the message body from a file option is used.

Example: Mailing an HTML File as Message Body

```
TABLE FILE file1
PRINT A B C
ON TABLE HOLD AS MYFILE FORMAT HTML
END
EX EDAMAIL  user1@corp1.com, user2@corp1.com, File1 Report,,HTML, MYFILE
```

Example: Mailing an HTML File as a Message Attachment

```
TABLE FILE file1
PRINT A B C
ON TABLE HOLD AS MYFILE FORMAT HTML
END
EX EDAMAIL  user1@corp1.com, user2@corp1.com, File1 Report,a,HTML, MYFILE
```

Example: Mailing a Multi-Line Inline Message

```
...
EX -LINES * EDAMAIL user1@corp1.com, user2@corp1.com, &SUBJECT,,,
    Run result for &TESTNAME is:
&RESULT
EDAMAIL*
```

Syntax: How to Use EX EDAMAIL in the Extended Form

The EDAMAIL extended form using name-value pairs is activated by the detection of an equal sign (=) in the first parameter. Parameter names are not case sensitive and may be in any order, but the message parameter (if used) must be last.

The syntax is:

```
EX EDAMAIL to=addresslist, [toaddr=addresslist,] [from=address,]
 [fromaddr=address,] [replyto=address,] [importance=low|normal|high,]
 [subject=string,] [flags=value,] [filetype=extension,]
 [filename=file,] [message=body message]
```

where:

`to=addresslist`

Is the email recipient displayed by the user email client (and used in an email client Reply To All, if a *toaddr* header is not supplied). Multiple addresses are allowed, using a semi-colon as the address separator. A comma is also allowed as a separator, if the overall address string is enclosed in single quotation marks.

If the *to* parameter is used in conjunction with *toaddr*, the value of *to* may be an arbitrary string, such as "Group Managers", which most email clients will display as a pseudo-title in the *To* field without displaying the actual addresses used in the *toaddr* parameter. A forced blank can be supped for the *To* field by using *to*="".

`toaddr=addresslist`

Is the email recipient. If not supplied, SMTP servers will use the *to* header. Email clients will use the *toaddr* value in an email client Reply To All. Multiple addresses are allowed, using a semi-colon as the address separator. A comma is also allowed as a separator, if the overall address string is enclosed in single quotation marks.

`from=address`

Is the email sender displayed by the email client (and used in an email client Reply, unless overridden by a *fromaddr* or Reply To header). If the *from* parameter is used in conjunction with *fromaddr*, the *from* value may be an arbitrary string, such as *The Boss*, which most email clients will display as a pseudo title in the From field, and will not display the actual address used in the *fromaddr* parameter.

`fromaddr=`*`address`*

Is the email sender. If not supplied, most email clients will use the From header when doing a reply.

`replyto=`*`address`*

Is the email sender. If not supplied, most email clients will use the *fromaddr* or *from* parameter value.

`importance=`*`low|normal|high`*

Is the email importance level for email clients that support importance flags. Valid values are high, normal, and low.

`subject=`*`string`*

Is the email subject string. If the subject string contains a comma, the string must be enclosed in single or double quotation marks.

`flags=`*`value`*

If set to a or A, the file is sent as an attachment. Otherwise, it is included as the body of the email.

`filetype=`*`extension`*

Defines the data file type for an email message body that uses a file as the body. Any application file type is valid, including MASTER, FOCEXEC, HTML, TEXT, and so on. Leave the parameter out to use the inline email message body feature.

`filename=`*`file`*

Defines the data file for an email message body that uses a file as the body. Leave the parameter out to use the inline email message body feature.

`message=`*`body message`*

Is the inline data stream containing the email message body. If used, it must be the last parameter in the EDAMAIL command. To use the inline data stream feature, the *filetype* and *filename* parameters cannot be supplied. The data stream may also be spread onto multiple lines if EDAMAIL is used with the EX -LINES {n|*} feature.

If an inline data stream message body is spread across multiple lines in the procedure, the resulting email is a single line of output. Multi-line message bodies are respected when the message body from a file option is used.

Example: Mailing an HTML File as a Message Body Using Multiple Addresses and Extended Form

```
TABLE FILE file1
PRINT A B C
ON TABLE HOLD AS MYFILE FORMAT HTML
END
EX EDAMAIL to=Managers,toaddr=user1@corp1.com;user2@corp1.com,
from=support1@corp1.com,subject=File1 Report, filetype=HTML,
filename=MYFILE
```

Configuring, Starting, and Stopping the FOCUS Database Server

To configure the FOCUS Database Server (sink machine or FDS), you must place configuration parameters in the odin.cfg communications file. The odin.cfg file is in:

install_dir/ibi/srv77/foc/home/etc

Reference: Configuration Parameters for the FOCUS Database Server

The following odin.cfg file configures the default names for the FOCUS Database Server. The default server node name is FOCSU. The default client node name is FOCSU01.

```
;FOCUS Database Server
NODE = FOCSU
BEGIN
  PROTOCOL = TCP
  CLASS = SUSERVER
  PORT = 8102
END

;FOCUS Database Client
NODE = FOCSU01
BEGIN
  PROTOCOL = TCP
  CLASS = SUCLIENT
  HOST = host
  PORT = 8102
END
```

where:

host
Is the host name or IP address for the FOCUS Database Server.

Reference: Starting and Stopping the FOCUS Database Server

To start the FOCUS Database Server, issue the focus command with the -sinkstart option:

```
focus -sinkstart [snodename]
```

where:

snodename
Is the server node name from the odin.cfg file. If you are using the default node name, it can be omitted.

To stop the FOCUS Database Server, issue the focus command with the -sinkstop option:

```
focus -sinkstop [cnodename]
```

where:

cnodename
Is the client node name from the odin.cfg file. If you are using the default node name, it can be omitted.

Reference: Configuring Multiple FOCUS Database Servers

Each instance of FOCUS can only have one FDS associated with it. In order to run multiple FOCUS Database Servers, multiple instances of FOCUS must be installed and started. Each instance of FOCUS needs its own set of port numbers, which are defined in the odin.cfg file.

Check with your Network Administrator to determine which port numbers you should use for additional sink machines.

One of the FOCUS instances will be considered the primary server and will be the one that the FOCUS Clients will use. The other server instances will be used only for the FDS component.

Running isetup and Creating New Directory Locations

To create the new FOCUS instance, run the isetup process and install the secondary FOCUS instance to a unique EDACONF name.

During the installation process you will be prompted to review the following default settings.

```
EDAHOME = …/ibi/srv82/foc/home
 EDACONF = …/ibi/srv7827/foc/wfs
 EDAPRFU = …/ibi/foc/profiles
 APPROOT = …/ibi/apps
 HTTP_BASE_PORT = 8121
```

where:

…/
References the absolute path where FOCUS was installed.

When asked to proceed with the defaults, enter *N* (for No) and edit the following directory locations:

1. Create a new EDAHOME for the new instance. For example:

```
/group1/user1/ibi/srv82/foc/home2
```

2. Create a new EDACONF for the new instance. For example:

```
/group1/user1/ibi/srv82/foc/wfs2
```

3. Create a new EDAPRFU for the new instance. For example:

```
/group1/user1/ibi/srv82/foc/profiles2
```

4. Enter the new port for the HTTP Listener. This is the second of six consecutive port numbers that must be open and available for FOCUS IP-based services.
5. If prompted for a mail server, press *Enter*, as this feature is optional.

Leave APPROOT and HOMEAPPS (if prompted) pointing to the APPROOT and HOMEAPPS path of your initial FOCUS installation.

Re-verify the configuration options, and if everything looks good, enter *Y* to *Accept and Proceed*.

Editing odin.cfg for the New Installation

Edit the secondary odin.cfg in ibi/srv77/foc/wfs2/etc. You only need to change the FOCUS Database Client name. The port numbers should already be set to the new port numbers configured during the installation process (Step 4). The FOCUS Database Server and Client sections should be similar to:

```
;FOCUS Database Server
NODE = FOCSU
BEGIN
  PROTOCOL = TCP
  CLASS = SUSERVER
  PORT = nnnn
END
;FOCUS Database Client
NODE = SINKB01
BEGIN
  PROTOCOL = TCP
  CLASS = SUCLIENT
  HOST = 127.0.0.1
  PORT = nnnn
END
```

Note that the FOCUS Database Server name is still FOCSU, but the port number is different from the primary server and the client node name is different.

Edit the secondary suprof.prf to allocate the files that will be used by this FDS, or use the APP PATH command to set the PATH where the alternate sink files are located.

Editing odin.cfg for the Original Installation

In the primary odin.cfg file for the original installation (in ibi/srvxx/wfs/etc/odin.cfg) you must add a FOCUS Database Client section that is identical to the FOCUS Database Client entry in the odin.cfg of the secondary server. This entry will point to your secondary (alterantive) sink:

```
;FOCUS Database Client
NODE = SINKB01
BEGIN
  PROTOCOL = TCP
  CLASS = SUCLIENT
  HOST = 127.0.0.1
  PORT = nnnn
END
```

Starting and Stopping the New Instance and the Secondary FOCUS Database Server

To start the secondary FOCUS instance and FDS, issue the following command:

```
ibi/srv82/foc/wfsalt/bin/focus -sinkstart
```

To stop the secondary FOCUS instance and FDS, issue the following command:

```
ibi/srv82/foc/wfsalt/bin/focus -sinkstop SINKB01
```

Chapter 6

z/OS Guide to Operations

As a z/OS user, you are familiar with the requirements of your particular operating system. These topics contain information about any FOCUS features that are unique to your system, as well as some proven methods for using FOCUS and allocating files in the z/OS environment.

All of the FOCUS features described in this documentation set are available to you.

In this chapter:

- Introduction to 64-bit FOCUS
- Referencing Files
- Application Files
- Window Files
- Extract Files
- Work Files
- Enabling Use of the zIIP Specialty Engine
- Calling FOCUS Under TSO
- FOCUS Facilities Under TSO
- TSO and FOCUS Interaction
- DYNAM Command

Introduction to 64-bit FOCUS

Starting with release 7709, FOCUS on z/OS is delivered as a 64-bit product. Some of the requirements specific to this version are:

- All of the data sets delivered are in extended format, including, for example, FOCEXEC and MASTER.

If you have your own versions of these data sets that you concatenate with the distributed libraries, you must either:

- Recreate them as PDSE libraries, if you want to concatenate them in front of the distributed libraries.
- Concatenate them following the distributed libraries.

- Compiled MODIFY procedures (FOCCOMP) are no longer supported. If you want to run a MODIFY procedure, you must have the MODIFY FOCEXEC available. The FOCUS RUN *focexec* command will now automatically be converted to EX *focexec*, so if you have a MODIFY FOCEXEC with the same name as the compiled MODIFY, it will be run.
- There is no longer the ability to create or use WINFORMS in conjunction with MAINTAIN procedures.
- You may need to recompile any user-written 3-GL subroutines you created, especially if they were created before FOCUS was required to be LE-compliant. Therefore, you must have the source code available.

Referencing Files

In FOCUS, you always reference files by DDNAME rather than by their fully qualified data set names. Data set names are arbitrary but must conform to your installation's naming standards.

Note: FOCUS supports up to 380 DDNAMEs and a total of 5000 members.

Reference: FOCUS Files

The following is a list of the major files that you will use in FOCUS. These will be discussed in detail in subsequent sections. The files are referenced by ddname and divided into categories:

Required DDNAMEs	Description
ERRORS	Contains error messages, help information, National Language error messages, README information, and FOCUS configuration parameters.
SYSPRINT	Specifies the normal destination of the run log, messages, and reports.
SYSIN	Is the source of the FOCUS command.

Required DDNAMEs	Description
OFFLINE	Specifies alternate destination for printed reports.

Application Files	Description
MASTER	Master Files.
ACCESS	Access Files (Optional except for intelligent partitioning. For information, see the *Describing Data* manual).
FOCEXEC	Stored procedures.
ddname	FOCUS data sources and external indices.
USERLIB	Library of user programs.
FOCSTYLE	FOCUS StyleSheet files.
ddname	Non-FOCUS data sources.
TTEDIT	TableTalk sessions.
FMU	Compiled Window files.
TRF	Documentation for window files or window transfer files.
HOLDSTAT Files	Documentation and DBA information for extract files.

Extract DDNAMEs	Description
*HOLD	Contains data saved using the HOLD command.
*SAVB	Contains data saved using the SAVB command.
*SAVE	Contains data saved using the SAVE command.

Extract DDNAMEs	Description
HOLDMAST	Temporary Master File for FOCUS HOLD files.

Note: There are also extract files that you must allocate if used: LET, LOG, POST, and Dialogue Manager output files.

Work Files	Description
STACK	Used by Dialogue Manager to store FOCUS commands.
FOCSORT	Used during sorting.
FOCSML	A work area used by the Financial Modeling Language.
*FOCPOST	Sequential output file saved using the POST or PICKUP commands.
REBUILD	Used by the REBUILD utility.
TABLTALK	Used by TableTalk as a procedure.

*The AS phrase renames asterisk-marked files to allow more than one file of that type in a single session.

Allocating Files

In order to run FOCUS, you must have a job, CLIST, or REXX EXEC that allocates the necessary files. The ISETUP installation procedure creates a startup job or CLIST for executing FOCUS. For information about ISETUP, see the *z/OS Installation Guide*.

You can also explicitly allocate files using JCL in your logon procedure or with TSO ALLOCATE, DYNAM ALLOCATE, or REXX commands. Additionally, FOCUS will dynamically allocate certain work and permanent files during a FOCUS session, if they conform to the standard naming conventions (see *Dynamically Allocating Files* on page 141 and *DYNAM Command* on page 200).

The JCL or CLIST needed to operate FOCUS is usually the same and varies only because of local installation conventions. Generally, file allocations are the same for all users and vary only for the particular data files referenced. FOCUS uses a standard set of DDNAMEs that perform specific functions. Their appearance is required in most runs.

Example: Allocating Files using JCL

The following is an example of JCL for a typical batch run:

```
//FOCUS EXEC PGM=FOCUS
//STEPLIB  DD DSN=FOCUS.FOCLIB.LOAD,DISP=SHR
//ERRORS   DD DSN=FOCUS.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//OFFLINE  DD SYSOUT=*
//MASTER   DD DSN=MASTER.DATA,DISP=SHR
//FOCEXEC  DD DSN=FOCEXEC.DATA,DISP=SHR
//* FOCUS  CAR DATABASE
//CAR      DD DSN=CAR.FOCUS,DISP=SHR
//SYSIN    DD *
 .
 . FOCUS commands
 .
FIN
/*
```

The JCL shown below is a typical TSO logon procedure for a FOCUS session:

```
//TSOLOGON EXEC PGM=IKJEFT01,DYNAMNBR=50
//* FOCUS DATASETS
//STEPLIB DD DSN=FOCUS.FOCLIB.LOAD,DISP=SHR
//ERRORS  DD DSN=FOCUS.ERRORS.DATA,DISP=SHR
//OFFLINE DD SYSOUT=A
//* USUAL TSO LOGON DATASETS FOLLOW
```

The following is an example of a TSO allocation CLIST:

```
ALLOC F(MASTER)   DA(MASTER.DATA)  SHR REUSE
ALLOC F(FOCEXEC)  DA(FOCEXEC.DATA) SHR REUSE
ALLOC F(CAR)      DA(CAR.FOCUS)    SHR REUSE
```

Note: These allocation procedures are discussed in more detail in *Calling FOCUS Under TSO* on page 179.

Dynamically Allocating Files

You do not have to explicitly allocate all of your files prior to using them in a FOCUS session. FOCUS will dynamically allocate certain files.

FOCUS will allocate some or all of the following output or work files as temporary data sets during a FOCUS session:

- HOLD, SAVB, and SAVE files.
- FOCUS data sources created by the CREATE FILE command.
- FOCUS work files such as STACK, FOCSORT, and HOLDMAST.

- Financial Modeling Language file FOCSML.
- OFFLINE, SYSIN, and SYSPRINT files.

FOCUS automatically allocates many permanent files on an as needed basis as long as they follow the data set naming conventions.

The permanent files that may be dynamically allocated are:

- Master Files
- Access Files
- FOCEXEC files
- Input data files
- TTEDIT files

Note: *Required Files* on page 142 through *Work Files* on page 166 contain summaries of the files used in FOCUS sessions, with descriptions of their functions and information about allocations. Review them carefully, keeping in mind the space requirements of your application. If the amount of space allocated automatically is insufficient, you must make your own allocation before attempting to use the file.

Required Files

The following files are required by FOCUS:

- ERRORS: This is the data set that contains the text of FOCUS and National Language error messages, the text printed in response to the HELP command, README information, and FOCUS configuration parameters (FOCPARM). This data set must be allocated prior to any FOCUS session. If ERRORS is not allocated, FOCUS will terminate with the following message:

```
(FOC000) UNABLE TO LOCATE ERRORS FILE, FOCUS PROCESSING IS
         UNPREDICTABLE ERROR READING FOCPARM INSTALL FILE.
```

- SYSPRINT: The normal destination of all FOCUS output is ddname SYSPRINT. In batch, this is usually a SYSOUT data set, but it can also be a sequential data set. This remains the destination of all FOCUS output unless you enter the FOCUS OFFLINE command to send the output to an offline printer or file. Sample allocations are:

```
//SYSPRINT DD SYSOUT=A
ALLOC F(SYSPRINT) DA(SYSPRINT.DATA)
ALLOC F(SYSPRINT) DA(*)
```

- SYSIN: The input file for FOCUS commands has the ddname SYSIN. It may be a data set, the card input stream, or the terminal itself. If the terminal is acceptable to you, you do not have to allocate SYSIN. Sample allocations are:

  ```
  //SYSIN DD *
  //SYSIN DD DSN=SYSIN.DATA,DISP=SHR
  ALLOC F(SYSIN) DA(*)
  ```

  Once you enter FOCUS, you should not reallocate SYSIN in the foreground.

- OFFLINE: If you enter the FOCUS OFFLINE command, FOCUS sends its report output to the destination specified by ddname OFFLINE, and messages to the destination specified by ddname SYSPRINT. FOCUS normally allocates this file for you when you enter the FOCUS environment. It specifies the OFFLINE destination as the printer for batch jobs and your terminal for TSO. You may close it for reallocation with the FOCUS command OFFLINE CLOSE.

You do not have to explicitly allocate SYSIN, SYSPRINT, and OFFLINE by means of JCL, ALLOCATE, or DYNAM ALLOCATE commands. However, for the OFFLINE file (OFFLINE default is the terminal), user allocation is normally specified.

Whether or not SYSPRINT or OFFLINE are allocated to the terminal, the active output width is controlled by the LINESIZE or SCRSIZE parameter of the TERMINAL command that is in effect at entry into FOCUS.

Caution: The characteristics of files SYSIN and SYSPRINT are tested only once, at entry into FOCUS. ALLOCATE, DYNAM ALLOCATE, or FREE commands affecting these files must not be issued from within FOCUS. Likewise, altering the terminal LINESIZE or SCRSIZE setting from within FOCUS is not recommended.

Application Files

This section describes how FOCUS refers to and searches for your application files, such as Master Files, FOCEXEC files (stored procedures), FOCUS data sources, USERLIB programs, non-FOCUS data sources, and HOLDSTAT files. It also describes how these files interact with one another under TSO, the various specifics regarding how these files are allocated, and their required DCB attributes.

Master Files

Master Files are partitioned data sets that contain Master Files, consisting of parameter lists that describe data sources to FOCUS. There are two types of Master Files:

- Permanent Master Files allocated to DDNAME MASTER.
- Temporary Master Files allocated to DDNAME HOLDMAST.

When FOCUS needs a Master File, it searches for the DDNAME MASTER. If FOCUS finds that the DDNAME MASTER is not allocated when a file is referenced, it attempts to allocate the data set '*prefix*.MASTER.DATA' to MASTER. If it does not find the member there, FOCUS next searches the DDNAME HOLDMAST. If it does not find the member in HOLDMAST either, FOCUS prints an error message.

The maximum LRECL for a variable length file is 32756 bytes; for a fixed length file, the maximum length is 32760 bytes. The record format can be fixed or variable. TED can work with files with an LRECL up to 32760.

Typical TSO and batch allocations for a Master File are:

```
ALLOC F(MASTER) DA(MASTER.DATA) SHR
//MASTER DD DSN=prefix.MASTER.DATA,DISP=SHR
```

The entire data set must be referenced in the allocation, not a particular member.

The data set member names must correspond to the FOCUS names of Master Files. For example, the FOCUS command TABLE FILE CAR requires a member CAR in the Master File data set.

Several partitioned data sets of FOCUS Master Files can be concatenated under the DDNAME MASTER, when they have identical LRECL and RECFM attributes. When the concatenation includes a mixture of variable and fixed record formats, the variable format must be listed first. The widest file must be on the top of the concatenation.

Optionally, you can number the records in a Master File in positions 73-80 for fixed block 80-byte files. In order to use numbers in these positions in RECFM=VB files, the numbers must be preceded by a ‘,$’ comment marker. FOCUS distinguishes between numbered and unnumbered data set members when the first record is read. If positions 73-80 are numeric, FOCUS assumes the file lines are numbered and only positions 1-72 are significant. However, FOCUS does not verify that the numbers are in ascending order.

To create and maintain Master Files under TSO, use the TED or ISPF editor. You can access the ISPF editor using the IEDIT command in FOCUS. In batch, use the utility programs IEBPTPCH and IEBUPDTE.

Access Files

Access Files for FOCUS data sources are members of partitioned data sets allocated to DDNAME ACCESS. They are optional except for intelligent partitioning of FOCUS data sources and multi-dimensional indexes. For information, see the *Describing Data* manual.

The maximum LRECL for a variable length file is 32756 bytes and for a fixed length file the maximum is 32760. The record format can be fixed or variable. TED can work with files with an LRECL up to 32760.

Typical TSO and batch allocations for an Access File are:

```
ALLOC F(ACCESS) DA(ACCESS.DATA) SHR
//ACCESS DD DSN=prefix.ACCESS.DATA,DISP=SHR
```

The entire data set must be referenced in the allocation, not a particular member.

The data set member names must correspond to the Master File member names. For example, the FOCUS command TABLE FILE CAR requires a member CAR in the ACCESS file data set.

When the Master File has been defined with a long member name, the actual member will be an eight-character name created from the initial characters of the long member name and a unique sequence of characters. For information, see the *Describing Data* manual.

FOCEXEC Files

All FOCUS procedures can be stored in one or more partitioned data sets allocated to DDNAME FOCEXEC. The names of the procedures correspond to the member names.

The maximum LRECL for a variable length file is 32756 bytes and for a fixed length file the maximum is 32760. The record format can be fixed or variable. TED can work with files with an LRECL up to 32760. The line numbering conventions for Master Files apply to FOCEXEC files as well (see *Master Files* on page 144).

You can use the TED editor to create FOCEXECs. However, the DCB attributes must have been provided before you use TED.

An alternative under TSO for creating FOCEXEC files is the ISPF editor. You can invoke the ISPF editor using the IEDIT command in FOCUS. In batch mode, you can create FOCEXECs with the utility programs IEBPTPCH and IEBUPDTE.

Syntax: How to Execute a Stored Procedure

```
EXEC procedurename
```

or

```
EX procedurename
```

where:

procedurename

Is the name of the procedure to be executed. This corresponds to the member name.

If you attempt to execute a procedure and no DDNAME FOCEXEC is found, FOCUS will try to issue the following allocation

```
ALLOC F(FOCEXEC) DA('prefix.FOCEXEC.DATA') SHR
```

where:

prefix

Is your TSO PROFILE PREFIX or the FOCUS SET PREFIX= value.

Several partitioned data sets of FOCEXEC procedures may be concatenated under the DDNAME FOCEXEC.

Example: Executing a Stored Procedure

Sample allocations for the FOCEXEC file in TSO and batch are:

```
ALLOC F(FOCEXEC) DA(FOCEXEC.DATA) SHR
//FOCEXEC DD DSN=prefix.FOCEXEC.DATA,DISP=SHR
```

PROFILE FOCEXEC

In FOCUS, the procedure named PROFILE must be either a member of the data set allocated to the DDNAME FOCEXEC or a separate file allocated to the DDNAME PROFILE. FOCUS checks the data sets allocated to the DDNAME FOCEXEC first, and then sequential files allocated to DDNAME PROFILE. Only the first PROFILE encountered is executed. You may also customize PROFILE procedures and specify these alternatives when you execute FOCUS, as described in *Calling FOCUS Under TSO* on page 179.

FOCEXECs as Sequential Files

If, when executing an EXEC statement, FOCUS does not find the procedure among the members of the data sets allocated to DDNAME FOCEXEC, FOCUS next checks for a sequential file, with a DDNAME that matches the procedure name in the EXEC statement. If such a file exists, FOCUS executes this file as the procedure. This feature enables you to store procedures in separate sequential files, which is particularly useful when you are developing and changing procedures frequently. Repeated editing of a PDS member contributes to the need to compress the PDS. This is not required for PDSE Libraries.

Note:

- Do not use the -RUN statement in FOCEXECs in sequential files, as -RUN closes physical sequential files. Thus, after execution, control will return to the beginning of the FOCEXEC instead of the line directly following the -RUN. (For more information on -RUN, see the *Developing Applications* manual.)
- FOCEXECs in sequential files may not contain system variables or TSO, -TSO,-INCLUDE, -GOTO, or -IF...GOTO statements. In general, a FOCEXEC in a sequential file should not contain any Dialogue Manager statements or variables because unpredictable results may occur.

Example: Allocating FOCEXECs as Sequential Files

Some typical allocations of sequential files used as FOCEXECs are as follows:

```
ALLOC F(TESTL) DA('TESTL.FOCEXEC.DATA') SHR
//TESTL DD DSN=TESTL.FOCEXEC.DATA,DISP=SHR
//TESTL DD *
.
.
FOCUS statements
/*
```

StyleSheet Files

All FOCUS StyleSheets can be stored in one or more partitioned data sets allocated to DDNAME FOCSTYLE. The record format can be fixed length or variable length with LRECL up to 32760 like FOCEXEC, MASTER, or ACCESS. For more information, see *Creating Reports*.

FOCUS Data Sources

FOCUS data sources contain data written in FOCUS format. See the *Describing Data* manual for information about maximum file size and partitioning. Each data source is allocated to a DDNAME that matches the member name of the file's Master File in the data set allocated to DDNAME MASTER. For example, if the data source's Master File has the member name LEDGER, then the data source is allocated to DDNAME LEDGER. You can override this default with the USE command, a DATASET attribute in the Master File, or an Access File. These techniques are explained in the *Describing Data* manual.

Allocating FOCUS Data Sources

FOCUS data sources are formatted in 4K pages. XFOCUS data sources are formatted in 16K pages. You can request both primary and secondary allocations. z/OS permits up to 15 extensions of secondary space which you can request in units of tracks, cylinders, or blocks. We strongly recommend that you allocate FOCUS data sources in cylinders, if the file is big enough to justify this.

FOCUS uses the primary space until it is exhausted, and then automatically expands the allocation one extent at a time as more disk storage space is needed for the data.

FOCUS supplies the DCB when you issue the CREATE command. For a FOCUS data source the DCB parameters are RECFM=F, LRECL=4096, and BLKSIZE=4096. For an XFOCUS data source, the LRECL and BLKSIZE are both 16384 (16K). You can use the FOCUS MODIFY and FSCAN commands to maintain these data sources.

If your site does not preallocate FOCUS data sources prior to using them, you can have FOCUS automatically allocate them by issuing the SET FOCALLOC command.

Example: Allocating a FOCUS Data Source

The following example is for a new FOCUS data source:

```
//CAR DD DSNAME=CAR.FOCUS,UNIT=SYSDA,DISP=(NEW,CTLG),
//VOL=SER=MYVOL,SPACE=(CYL,(5,5))
```

Syntax: How to Set FOCUS to Automatically Allocate FOCUS Data Sources

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

where:

`ON`

Causes FOCUS to automatically allocate FOCUS data sources with a data set name of *prefix.masterfile*.FOCUS.

`OFF`

Indicates that you must allocate your FOCUS data sources. This is the default value.

Multi-Volume Support

You have the option of allocating new FOCUS data sources, XFOCUS data sources, and FOCUS-created sequential files across multiple volumes.

Note: If your site has SMS rules that prohibit multi-volume allocations, the SMS rules will be respected.

Many sites prefer to distribute high volume data sources across multiple volumes in order to manage:

- Use of storage on specific devices or device types.
- Run-time access to these devices.

You can use this performance tuning technique (also known as *data striping*) with FOCUS data sources, XFOCUS data sources, and FOCUS-created sequential files in the z/OS batch, and TSO environments.

The SPACE parameter for allocating any data source can include a primary and a secondary allocation. The primary allocation is the amount of space allocated the first time data is written to the data set. The secondary allocation is the amount of space to be allocated, when necessary, for up to 15 additional extents.

For a single-volume data source, processing terminates with a B37 abend when the system detects either of the following conditions:

- A need for more than 15 extents.
- A need for a new secondary extent (below the 16-extent limit) when enough space is not available on the volume.

FOCUS returns the following message to indicate that one of these conditions has occurred:

```
(FOC198) FATAL ERROR IN DATABASE I/O. FOCUS TERMINATING CODE: 00000070
```

You can prevent this type of abnormal termination by allocating multiple volumes to the data source. With multiple volumes, an out of space condition on the first volume causes allocation to start on another volume.

With multiple volumes, the allocation process varies slightly for each of the following:

- First volume
- Intermediate volumes
- Last volume

The following table describes the multi-volume allocation process:

Primary Allocation	The primary allocation is applied to the first volume only. It can consist of the number of extents allowed by z/OS for a primary allocation. **Note:** A data source with no secondary space allocation is limited to a single volume.

Secondary Allocation	1. **First volume.** As many extents as are available, up to the 16-extent limit, are allocated and filled before continuing to the second volume.
	2. **Intermediate volume.** Depending on the space available, up to 16-extents are filled before allocation begins on the next volume.
	3. **Last volume.** Once the need for a number of extents greater than the limit is detected:
	❑ For a FOCUS data source, processing terminates with the following message:
	`(FOC198) FATAL ERROR IN DATABASE I/O. FOCUS TERMINATING CODE 00000070`
	❑ For temporary FOCSORT files, after all volumes and extents are filled, allocation spills to up to 15 additional temporary files, each with 16 extents. The SPACE allocation for each spill file is the same as the SPACE allocation for the original FOCSORT file. For example:
	`//FOCSORT DD SPACE`
	This allocates a total of 5 + (5*15) = 80 tracks. When the 81st track is needed, another temporary data set is allocated with the parameter SPACE=(TRK,(5,5)). If necessary, this additional step is repeated a total of 15 times yielding a total of 80*16 tracks for FOCSORT.
	If enough space is not available after filling all of the extents of all of the spill files, the FOC198 message is issued and processing terminates. FOCUS places no limits on the size of the FOCSORT file. The limit is determined by the operating system and the amount of space available. You should estimate your space requirements and set your primary and secondary allocations accordingly.
	Note: The actual number of extents actually obtained on any volume may be less than 16; however, in most situations 16 will available and used.

Syntax: **How to Allocate a Multi-Volume Data Source in the z/OS Batch Environment**

You have two choices for statically allocating a new multi-volume FOCUS data source, XFOCUS data source, or sequential file.

You can list multiple VOLSER identifiers on the DD card for the multi-volume data source:

```
//ddname DD DSN=dsname,VOL=SER=(vol1,...,voln),...
```

Alternatively, you can ask for multiple units of a specific type

```
//ddname DD DSN=dsname,UNIT=(type,n),...
```

where:

ddname

Is the DDNAME associated with the multi-volume data source.

dsname

Is the data set name of the multi-volume data source.

vol1,...,voln

Are the volume identifiers for each of the volumes to use.

type

Is the type of unit to use.

n

Is the number of units.

Allocating a Multi-Volume Data Source in TSO and z/OS FOCUS

In both TSO and z/OS FOCUS you have two choices for dynamically allocating a multi-volume data source:

- You can list multiple volume identifiers.
- You can specify the number of units to use and let the system choose the specific volumes. All of the units will be the same type (for example, 3390).

Syntax: **How to Allocate Specific Volumes in TSO and z/OS FOCUS**

To allocate specific volumes for a multi-volume data source, use the following syntax:

In TSO

```
TSO ALLOC ... VOLUME('vol1,...,voln')...
```

In z/OS FOCUS

```
DYNAM ALLOC ... VOL vol1,...,voln ...
```

where:

vol1,...,voln

Are the volume identifiers for the each of the volumes to use.

Syntax: **How to Specify the Number of Units in TSO and z/OS FOCUS**

To specify the number of volumes for a multi-volume data source and let the system choose the specific volumes, use the following syntax:

In TSO

```
TSO ALLOC ... UCOUNT('n') UNIT('type') ...
```

In z/OS FOCUS

```
DYNAM ALLOC ... UCOUNT n UNIT type ...
```

where:

n

Is the number of volumes to use.

type

Is the type of unit to use.

Note:

- UNIT VIO is not supported.
- The RLSE option of the SPACE parameter is not supported.

Example: **Allocating a Data Source to Two Volumes**

The following DYNAM command allocates two volumes to a data source called MULTVOL:

```
DYNAM ALLOC FI MULTVOL DS USER1.FOCTST.MULTVOL TRACK SPACE 4 4 REU -
 UCOUNT 2 UNIT SYSDA CATALOG
```

With this allocation, a second volume will be used when the 17th extent is needed.

Syntax: How to Display the Volume Identifiers Allocated to a Multi-Volume Data Source

To see the data set information associated with a specific DDNAME, issue the following command

```
? TSO DDNAME ddname
```

where:

ddname

Is the DDNAME allocated to the data set whose volume identifiers you want to see.

Example: Displaying Multi-Volume Data Set Information

The following example shows how to display data set information for DDNAME MULTVOL:

```
? TSO DDNAME MULTVOL
```

The following information is returned. Notice that two volume serial identifiers are listed on the VOLSER line:

```
DDNAME      =  MULTVOL
DSNAME      =  USER1.FOCTST.MULTVOL
DISP        =  NEW
DEVICE      =  DISK
VOLSER      =  MFOC02,MFOC01
DSORG       =  PS
RECFM       =  F
SECONDARY   =     4
ALLOCATION  =  TRACKS
BLKSIZE     =      4096
LRECL       =      4096
TRKTOT      =        92
EXTENTSUSED =        23
BLKSPERTRK  =        12
TRKSPERCYL  =        15
CYLSPERDISK =      2227
BLKSWRITTEN =      1104
>  >
```

External Indices for FOCUS Data Sources

An external index is a FOCUS file that contains index, field, and segment information for one or more specified FOCUS data sources. The external index is independent of its associated FOCUS data source and is used to improve retrieval performance.

In z/OS, the external index is automatically allocated as a temporary file when it is created using REBUILD. To create the permanent external index, you must allocate it before you issue the REBUILD command.

SORTOUT is a work file that is used during the process to create an external index. You must allocate the work file with the proper amount of space, minus DCB parameters. To estimate the amount of space, use this formula:

```
bytes= (field_length + 20) * number_of_occurrences
```

MDIs for FOCUS Data Sources

An MDI is a separate multi-field index file for one or more FOCUS databases. The MDI is independent of its associated FOCUS database and is used to improve retrieval performance.

To create the MDI, you must allocate it before issuing the REBUILD command. The DCB attributes are RECFM=F, LRECL=4096, BLKSIXE=4096.

Disposition of FOCUS Data Sources

Existing FOCUS data sources can always be allocated as SHR by both TSO users and background jobs, even when being modified. Concurrent destructive updates are prevented by FOCUS itself. FOCUS reserves exclusive use of the data source, for update purposes, for the duration of the MODIFY command. At the conclusion of the MODIFY, the TSO user or batch job that had possession of the data source relinquishes control, making the data source available to the other users.

While only one TSO user or batch job is allowed to update the data source, multiple users may have the data source open concurrently for read-only purposes. This scheme offers the same protection as exclusive allocation (DISP=OLD), but it is more flexible, because with DISP=OLD, the data source is reserved for the duration of the allocation (which, for a batch job, is the entire time from initiation to termination).

If a TSO user attempts to modify a FOCUS data source currently being modified by another user or batch job, or if the data source is controlled by a FOCUS database server (sink machine), the result is a message stating that the data source is in use elsewhere and the MODIFY command flushes to the END statement allowing you to perform other tasks. On the other hand, if a batch job encounters the same situation, it will wait until the data source is free (that is, until the MODIFY currently in control ends or the FOCUS database server is terminated).

When TABLE and TABLEF commands are run for the purpose of locating cross-referenced segments (segment types KU and KM), you must allocate the data source with DISP=OLD or NEW. This allows for their addresses to be written into the data source from which they are being cross-referenced. This process, called pointer resolution, does not take place if the disposition of the data source is SHR or if the data source resides on a FOCUS database server.

Database Security: ENCRYPT, DECRYPT and RESTRICT

Since the restriction information for a FOCUS data source is stored in its Master File, you will want to encrypt the Master File in order to prevent users from examining the restriction rules. Only the Database Administrator can encrypt a Master File. If you wish to change restrictions, the process can be reversed using the DECRYPT command to restore the Master File to a readable form. You can add security limitations to existing data sources using the RESTRICT command.

Syntax: How to ENCRYPT a FOCUS File

```
ENCRYPT FILE membername {MASTER|FOCEXEC}
```

where:

membername

Is the name of the member in the data set to be encrypted.

Example: Encrypting a FOCUS File

The following command encrypts member EMPLOYEE of the data set allocated to DDNAME MASTER:

```
ENCRYPT FILE EMPLOYEE
```

Syntax: How to DECRYPT a FOCUS File

```
DECRYPT FILE membername {MASTER|FOCEXEC}
```

where:

membername

Is the name of the member of the data set to be decrypted.

Syntax: How to RESTRICT a FOCUS File

```
RESTRICT
ddname1
ddname2
.
.
.
END
```

where:

ddname1, ddname2

Are the DDNAMEs allocated to the files that will be restricted.

Example: Restricting a FOCUS File

The following restricts the files allocated to DDNAMEs CAR and EMPLOYEE:

```
RESTRICT
CAR
EMPLOYEE
END
```

FOCUS Data Sources and IBM Utility Programs

FOCUS data sources appear to the operating system as simple sequential data sets with fixed-length, unblocked 4096-byte records. Thus, FOCUS data sources can be processed by all of the common IBM data set utility programs (such as IEBGENER and the TSO COPY command). Every IBM utility program that works on sequential files (DSORG=PS) may be used, so you can freely move, back up, copy, and store FOCUS data sources in whatever way your installation chooses.

USERLIB

FOCUS searches for all dynamically loaded programs in two program DDNAMEs, USERLIB and STEPLIB. USERLIB is searched first, if allocated. If the desired program is not found in the USERLIB DDNAME, then the call library, STEPLIB, JOBLIB, link pack area, and linklist are searched.

If FOCUS is running APF Authorized, data sets allocated to USERLIB must be authorized.

USERLIB, if needed, must be allocated before its first use. This can be done in the logon procedure, through an ALLOCATE command, from within FOCUS, or without. However, if issued from within FOCUS, it cannot be reallocated after the first use. The allocation remains in effect for the remainder of the FOCUS session.

If the special function library called *myprogram.load* is needed, it should be allocated to DDNAME USERLIB. For example:

```
ALLOC F(USERLIB) DA('myprogram.load') SHR
```

Any arbitrary library is allocated as USERLIB:

```
ALLOC F(USERLIB) DA('USER.LIBRARY.load') SHR
```

These libraries can be concatenated as follows:

```
ALLOC F(USERLIB) DA('myprogram.load USER.LIBRARY.load') SHR
```

Note: Within FOCUS, the TSO ALLOCATE or DYNAM ALLOCATE command must be used.

Window Files

Window files contain the windows, menus, and related information created by Window Painter. There are three types of window files:

- Compiled window files are created when a Window Painter user chooses the Create a new file option, or invokes Window Painter and specifies a window file that does not exist yet. Compiled window files are also created when a window transfer file is compiled by the WINDOW COMPILE command. Compiled window files can be executed by a Dialogue Manager -WINDOW statement, and can be edited using Window Painter.

- Window transfer files are created by selecting the Create a transfer file option from the Window Painter Utilities Menu. Transfer files are uncompiled source code versions of compiled window files; they can be transferred from FOCUS running in one operating environment (for example, z/OS) to FOCUS running in another operating environment (for example, UNIX), and then edited to remove or fine tune window features not fully supported in the new environment. Transfer files can be edited using TED or IEDIT. Before they can be executed efficiently by a -WINDOW command or edited by Window Painter, they must be compiled using the WINDOW COMPILE command.
- Window documentation files are created by selecting the Document a file option from the Window Painter Utilities Menu. A documentation file provides a window application developer with detailed information about the windows in a given window file. Documentation files can be edited using TED or IEDIT.

Compiled Window Files

Compiled window files are members of a data set. Before they can be created by Window Painter or by the WINDOW COMPILE command, a data set must be created with LRECL 4096 and RECFM FB, and allocated to DDNAME FMU.

Once the data set is allocated, Window Painter and the WINDOW COMPILE command will create compiled window files as required. The member name will be the window file name specified in the Window Painter session or the WINDOW COMPILE command.

Note that creating the data set is not necessary if you are creating window files to be used only in the same FOCUS session. Window Painter will temporarily allocate the data set.

Window Transfer Files and Window Documentation Files

Window transfer files and window documentation files are both members of the same data set. Before they can be created by Window Painter, the data set must be created with LRECL 80 to 132 and RECFM FB, and must be allocated to DDNAME TRF.

Once the data set is allocated, Window Painter can create window transfer files and window documentation files as needed, and transfer files can be transferred from FOCUS running in other environments as needed. Note that before transferring a file from another operating environment, you will need to allocate a new member in the TRF data set for it.

The member name for documentation files will be the window documentation file name specified in the Window Painter session. The member name for transfer files will be the window transfer file name specified in the Window Painter session, or the member name specified in the TSO ALLOCATE or DYNAM ALLOCATE command issued prior to the transfer.

Note: Creating the data set is not necessary if you are creating window transfer files and documentation files to be used only in the current FOCUS session. Window Painter will temporarily allocate the data set.

If you need to create both a documentation file and a transfer file for a given compiled window file, be sure to create these two files with different names; otherwise, the newer one will overwrite the older one in the data set.

Non-FOCUS Data Sources

You can use the FOCUS query language to read non-FOCUS data sources. FOCUS can read QSAM, ISAM, VSAM files; IMS, CA-IDMS/DB, ADABAS, MODEL 204, Millennium, DB2, CA-DATACOM/DB, and Teradata tables.

The DDNAME under which you should allocate the non-FOCUS data source depends on the type of data source. You should allocate QSAM, ISAM, and VSAM files to a DDNAME that corresponds to a member name in the data set allocated to DDNAME MASTER. Database files have their own rules governing DDNAMEs.

Except with QSAM files, FOCUS needs an Interface program to read non-FOCUS data sources. To read some IMS, CA-IDMS/DB, and ADABAS database files, you must also allocate all of the files normally used with these data sources.

Syntax: How to Set the VSAM Addressing Mode

```
MVS VSAM SET AMODE 31
```

With AMODE 31, FOCSAM builds ACBs and buffers in 31-bit addresses. 31 is the default.

To determine the addressing mode that is in effect at any time, you can issue the query

```
MVS VSAM SET ?
```

which returns the following output:

```
(FOC1177)    SET OPTIONS - : BUFND = n / BUFNI = n / AMODE = n
```

TTEDIT Files

When you save a TableTalk session, a command file and a session file are saved. These files are written as follows:

- The command file is written as a member in the data set allocated to DDNAME FOCEXEC if it is allocated OLD or NEW. If FOCEXEC is allocated as SHR or is not allocated, the command file is written to the data set '*prefix*.FOCEXEC.DATA'. If there is no '*prefix*.FOCEXEC.DATA', the commands are written to a sequential data set.

- The TableTalk session is written as a member in the data set allocated to DDNAME TTEDIT if it is allocated OLD or NEW. If TTEDIT is allocated as SHR or is not allocated, the session file is written to the data set '*prefix*.TTEDIT.DATA'. If there is no '*prefix*.TTEDIT.DATA', the session is written to a sequential data set; the session will only be available for editing by exiting TableTalk, re-entering TableTalk, and selecting *Last Request*.

The FOCEXEC and TableTalk session are saved in their respective data sets as the member name you specify in TableTalk.

To save and edit TableTalk sessions, allocate a data set to DDNAME TTEDIT as OLD. An allocation for TTEDIT is not needed if the data set '*prefix*.TTEDIT.DATA' exists. The data set should have LRECL=80,RECFM=FB.

If you omit the file name, a list of all TTEDIT files will be displayed. Include a file name to edit a particular saved TableTalk session.

Syntax: How to Edit a Saved TableTalk Session

To edit saved TableTalk sessions, enter the command

```
TABLETALK EDIT [filename]
```

where:

filename

Is the file name you specified in the saved TableTalk session. If you omit the file name, a list of all TTEDIT files will be displayed. Include a file name to edit a particular saved TableTalk session.

HOLDSTAT Files

HOLDSTAT files enable you to include FOCUS DBA information and environmental comments in HOLD and PCHOLD Master Files. Member HOLDSTAT in the distributed library 'prefix.ERRORS.DATA' is the default. Alternately, you may create your own HOLDSTAT or another user-specified member in your ERRORS or MASTER data sets.

The contents of a HOLDSTAT file are automatically included in HOLD and PCHOLD Master Files when the SET HOLDSTAT command is specified to ON or to a member name. For information about the SET HOLDSTAT command, see the *Developing Applications* manual.

Syntax: How to Specify a HOLDSTAT File

The HOLDSTAT file may exist as a member in the ERRORS or MASTER library:

```
'prefix.ERRORS.DATA{(HOLDSTAT)|(membername)}'
```

or

```
'prefix.MASTER.DATA{(HOLDSTAT)|(membername)}'
```

where:

prefix

Is the high-level qualifier used at your site.

membername

Is the member name of your customized HOLDSTAT file.

Note: When FOCUS searches for the HOLDSTAT file, the MASTER DDNAME takes precedence over the ERRORS DDNAME.

A HOLDSTAT file may contain environmental comments like a file header, or the FOCUS DBA attribute, or both. The supplied HOLDSTAT member contains the following file header with Dialogue Manager system variables:

```
$================================================================$
$      HOLD file created on &DATE at &TOD by FOCUS &FOCREL       $
$              Database records retrieved= &RECORDS              $
$              Records in the HOLD file = &LINES                 $
$================================================================$
```

In the HOLD Master File, the comments appear after the FILE and SUFFIX attributes and the DBA information is appended to the end.

Reference: Considerations for Creating a HOLDSTAT File

If you create your own HOLDSTAT file, consider the following rules:

- Each line of comments must begin with a dollar sign ($) in column 1.
- Comments may not include user-defined variables.
- List the DBA information after any comments. On separate lines, specify the keywords $BOTTOM and END, beginning in column 1, followed by the DBA attribute. The syntax is:

  ```
  $BOTTOM
  END
  DBA=password,$
  ```

  You may include other DBA attributes such as USER, ACCESS, RESTRICT, NAME, and VALUE. For information about the DBA attributes, see the *Describing Data* manual.

Extract Files

Extract files save lines of user output generated during a FOCUS session. If you do not allocate these files, FOCUS allocates HOLD, SAVE, SAVB, and HOLDMAST files. You must explicitly allocate the files described in *Extract Files That You Allocate: LET, LOG, POST, Dialogue Manager Output Files* on page 165.

HOLD Files

A HOLD file is a sequential data source that contains the results of a report request. To save report output in a HOLD file, you must execute the HOLD command.

Data for a temporary HOLD file is written to a data set whose DDNAME is HOLD or was specified with an AS phrase. This data set is a sequential file. FOCUS provides the DCB parameters in accordance with the record length of the report it is about to store. The DCB BLKSIZE parameter is automatically calculated. (To change this default, specify the SET BLKCALC command as described in the *Developing Applications* manual.) The layout of the HOLD file can be obtained from within FOCUS by issuing the FOCUS ? HOLD command.

Typical allocations for a HOLD file in TSO and batch are:

```
ALLOC F(HOLD) SP(20 10) CYL
//HOLD DD UNIT=SYSDA,SPACE=(CYL,(20,10))
```

If you have FOCUS store another report in the same HOLD file, it will overwrite the previous one, and FOCUS will assign new DCB attributes in accordance with the new report's records. For example, if you specify HOLD AS MASTER, it will overwrite your Master File. (If you do want to save the file, we recommend you make it part of the standard FOCUS CLIST and allocate it as a permanent data set.)

Syntax: How to Create a HOLD File

To save report output in a HOLD file, execute the HOLD command

```
HOLD [AS ddname]
```

where:

ddname

Is the sequential data set in which FOCUS saves the report output. FOCUS stores a Master File describing the report output in a member of the temporary data set allocated to DDNAME HOLDMAST. If you omit the *AS ddname* option, the DDNAME defaults to HOLD. The default allocation is five primary and five secondary cylinders.

SAVB Files

A SAVB contains the results of a report request with all numeric report fields in binary format. The file cannot be printed. Also, all character fields are padded with spaces to a multiple of 4 bytes.

Syntax: How to Create a SAVB File

```
SAVB [AS ddname]
```

where:

ddname

Is the name under which FOCUS allocates a temporary sequential data set.

FOCUS dynamically allocates a temporary sequential data set under DDNAME SAVB or a DDNAME you supply with the *AS ddname* option. FOCUS allocates five cylinders for primary space and 10 cylinders for secondary space. Record format is variable blocked with record length and block size dependent on the record size. Once DCB attributes are assigned, they remain in effect for the duration of the session, even if another SAVB is issued for the same file. To keep the file, use the DYNAM COPY command.

SAVE Files

A SAVE file contains the results of a report request with the external character format equivalent to SAVB. The command format and allocations are the same as SAVB. However, the numbers are printable EBCDIC characters, and no padding takes place.

Syntax: How to Create a SAVE File

```
SAVE [AS ddname]
```

where:

ddname

Is the name under which FOCUS allocates a temporary sequential data set.

Temporary Master Files: HOLDMAST Files

When FOCUS HOLD files are created, either under the default name HOLD or under a specified name (for example, HOLD AS MYNAME), the description is written into the data set whose DDNAME is HOLDMAST. This data set is exactly like a MASTER data set except that FOCUS creates the members.

Note: If you are using an adapter to create a relational table using HOLD, FOCUS also creates an Access File allocated to DDNAME HOLDACC. The following rules for HOLDMAST also apply to HOLDACC.

If HOLDMAST is not allocated when a HOLD file is created, FOCUS will allocate HOLDMAST as a partitioned data set with five primary and 10 secondary tracks.

Whenever FOCUS needs the description of a data source, it first searches DDNAME MASTER. If the member is not found, it then searches DDNAME HOLDMAST.

Typical allocations for a temporary HOLDMAST file in TSO and batch are:

```
ALLOC F(HOLDMAST) SP(1 1) TRACK DIR(1)
//HOLDMAST DD UNIT=SYSDA,SPACE=(TRK,(1,1,1))
```

DCB parameters must not be supplied by the user. FOCUS will create the HOLDMAST file with the current DCB. Members of the HOLDMAST file are created without line numbers.

If you want to retain the HOLDMAST file, give it a name and a DISP parameter.

Extract Files That You Allocate: LET, LOG, POST, Dialogue Manager Output Files

FOCUS does not automatically allocate the following files. You must allocate them and supply the appropriate DCBs:

- **LET Files.** To save your LET statements in a file, issue the command:

  ```
  LET SAVE [membername]
  ```

  FOCUS will store your LET statements in *membername* as a FOCEXEC. The default member name is LETSAVE. If FOCEXEC is not allocated, FOCEXEC is automatically allocated to '*userid*.FOCEXEC.DATA', and the LETSAVE member is saved there.

- **LOG Files.** If you want to keep one or more files listing the transactions processed by the MODIFY command, issue the subcommand

  ```
  LOG type ON ddname
  ```

  where:

  type

  Is the error type criteria. FOCUS stores the list in a file allocated to DDNAME. There is no default for the DDNAME. DCB attributes must be provided for this DDNAME to correspond with the records being written.

- **POST Files.** If, when using Financial Modeling Language (FML), you want to save an intermediate result generated by a report, issue the following command:

```
POST [TO ddname]
```

FOCUS will store the results in a file allocated to *ddname*. If you omit the TO *ddname* option, the DDNAME defaults to FOCPOST.

- **Dialogue Manager Output Files.** The Dialogue Manager command -WRITE *ddname* allows you to send messages to the terminal if *ddname* is SYSIN or to the file allocated to *ddname*. There is no default for *ddname*.

You can allocate these files to multiple volumes. See *Multi-Volume Support* on page 148 for information.

Work Files

In order to process certain types of requests, FOCUS needs to create temporary work files. Default allocations vary depending on the type of work file. See the *z/OS Installation Guide* for details.

STACK

Dialogue Manager uses a memory stack for work space with a default size of 8K bytes. However, if this is exceeded, FOCUS allocates a work file with ddname STACK as a temporary data set.

FOCSORT

The TABLE, TABLEF, GRAPH, and MATCH commands may require a sort work file. This is allocated under DDNAME FOCSORT when required. If FOCSORT is allocated to a sequential file, FOCUS respects the allocation.

The FOCSORT file can grow to any size allowed by the operating system running and the available disk space. The user does not have to break a request up to accommodate massive files. With no limit enforced by FOCUS, the operating system provides whatever warning and error handling it has for the management of a FOCSORT file that exceeds its limits.

If your original allocation is too small to accommodate the records retrieved, FOCUS allocates up to 15 additional extents based on the space attributes of the original allocation. For very large reports, more space may be required (the default is 5,5 cylinders). Allocate additional space as follows:

```
ALLOC F(FOCSORT) SPACE(10,10) CYLINDERS UNIT(SYSDA)
```

Note: Before you use the COMBINE command, FOCSORT must be allocated to the FOCUS database server.

FOCSML

When Financial Modeling Language is used, a work area with ddname FOCSML is automatically allocated by FOCUS when the FML command FOR is used.

FOCPOST

If Financial Modeling Language options POST or PICKUP are used, a sequential output file must be allocated under ddname FOCPOST:

```
ALLOC F(FOCPOST) SP(1 1) TRACKS
//FOCPOST DD SPACE=(TRK,(1,1)),UNIT=SYSDA
```

REBUILD

The REBUILD command options REBUILD and REORG require that a work file be allocated under DDNAME REBUILD. (See the *Maintaining Databases* manual for detailed information on the REBUILD command.) FOCUS allocates this file as a data set when you execute the REBUILD command for options REBUILD and REORG. All REBUILD options require a Master File for the data source being rebuilt, a data source to process, and a REBUILD work file. In addition, the REORG load phase requires a new data file for the reorganized FOCUS data sources. The INDEX option requires you to allocate the following SORT files: SORTIN, SORTOUT, and SYSOUT.

Sample allocations are illustrated below:

```
TSO FREE  F(SORTIN SORTOUT SYSOUT)
TSO ALLOC F(SORTIN)  SP(5 5) TRACKS
TSO ALLOC F(SORTOUT) SP(5 5) TRACKS
TSO ALLOC F(SYSOUT)  DA(*)
```

File SYSOUT is for critical sort error messages and, if allocated to a disk file, needs only a minimum amount of space (1 track).

FOCUS automatically allocates SORTWORK files. The space requirements of the work files SORTWK01 through SORTWK06 depend on the volume of data being sorted. If your application requires more SORTWORK space than FOCUS allocates, you must allocate these files with the necessary space parameters.

SORTIN and SORTOUT have identical space requirements which can be estimated very accurately, as follows:

- Records are fixed blocked, 100 to a block.
- Each record consists of a value of the field being indexed, rounded up to a 4-byte multiple, plus 8 bytes. The minimum record length is 18 bytes.

- The number of records can be obtained in response to the ? FILE file name command. It is the number of active instances of the segment in which the field to be indexed resides.

All must be allocated without DCB parameters.

You can only REBUILD one portion of a partitioned data source at one time. You must explicitly allocate the file containing the partition to use in the REBUILD request. You should REBUILD/REORG to a new file in sections, to avoid having to allocate large amounts of space to REBUILD. To do this, in the dump phase use selection criteria to dump a section of the data source. In the LOAD phase, make sure to *add* each new section after the first. To add to a data source, you must issue the LOAD command with the following syntax:

```
LOAD NOCREATE
```

TABLTALK

This file is used for storing the procedure to be executed from within TableTalk. This is a sequential file, dynamically allocated, used by TableTalk.

Enabling Use of the zIIP Specialty Engine

If your site has a zIIP (System **z** **I**ntegrated **I**nformation **P**rocessor) specialty engine from IBM, you can offload specific categories of workload from the Central Processors to the zIIP.

The zIIP engine is a restricted version of a Central Processor (CP), also referred to as a General Processor (GP). The capacity of the zIIP engine does not count toward the overall MIPS rating of the mainframe image, so the CPU usage incurred on the zIIP is effectively free. Central Processors are often configured to run at speeds below their maximum rating for cost saving and capacity planning purposes. For Central Processors, *100% capacity* typically refers to the maximum MIPS that the processor is allowed to generate at that installation, in accordance with your contract with IBM. In contrast, the zIIP engine always runs at true 100 percent of capacity.

As much as 80 percent of FOCUS processing is enabled to run on the zIIP engine. Typical workloads are expected to offload 30 to 80 percent of CPU processing to the zIIP engine.

To make use of the zIIP enablement feature, FOCUS must run in an authorized state. In some customer environments, certain processes are not allowed to run in an authorized state. If you attempt to run such a process in an authorized state, FOCUS will abend. In those situations, you must call FOCUS from a separate non-authorized environment.

What Is a zIIP Specialty Engine?

Though physically identical to a Central Processor, the zIIP engine is microcoded at installation time to run specific types of workloads. The Central Processor continues to handle the operating system, I/O interrupts and timer interrupts, job initiations, and user interactions with the operating system. The zIIP concentrates on CPU intensive workloads, leaving the Central Processor more time to absorb otherwise queued workloads, thereby achieving some overall performance improvement across all mainframe activity.

Steps to FOCUS zIIP Enablement

This section describes steps and requirements for FOCUS use of the zIIP processor.

The steps to FOCUS zIIP enablement are:

1. Obtain a personalized FLICENSE file that licenses you to use the zIIP feature. For instructions, see the *z/OS Installation Guide*.
2. Obtain APF authorization for all load libraries. For instructions, see *Obtaining APF Authorization for All Load Libraries* on page 170.
3. For Interactive processing, additionally obtain AUTHCMD and AUTHPGM authority. For instructions, see *Obtaining AUTHCMD and AUTHPGM Authority for TSO Processing* on page 172.
4. Activate the zIIP feature using the SET ZIIP=ON or SET ZIIP=ON/SIMMAXZIIP command. For instructions, see *Activating a zIIP Environment or Projecting zIIP Usage* on page 172.

Your site may not allow certain processes to run in an authorized state. Therefore, there may be times when you need to run FOCUS in an unauthorized state. For information about establishing a non-authorized environment after enabling the zIIP feature, see *Establishing a Non-authorized Environment After Enabling the zIIP Feature* on page 175.

Reference: Usage Notes for Use of the zIIP Processor

- Enabling the zIIP processor turns of the use of the external sort (SET EXTSORT=OFF).

  You can turn the external sort back on after zIIP has been enabled, if required. However, due to substantial sorting performance improvements confirmed by benchmark tests, using the FOCUS sort with the zIIP may be a better solution than moving to the external sort tool.

- IBM restricts the DB2 DSN command to TMP tasks only. This means that a call to DB2 using the DSN command will not create the authorized environment that is required to invoke an Enclave SRB transaction. Therefore, DSN CMD transactions are not zIIP eligible. This is a DB2 restriction that IBM does not plan to change. Therefore, the following sample JCL for calling FOCUS from DB2 is not zIIP-eligible:

```
//SYSTSIN  DD *
DSN SYSTEM (DS0)
RUN PROGRAM(FOCUS) PLAN(DSQLTA)
END
/*
```

- Maximize the blocksizes of data sources that are read or written by FOCUS to reduce the number of I/Os required to access the file. This will reduce the number of switches to non-zIIP mode that FOCUS has to make, thus permitting a greater percentage of zIIP contribution to the request.
- Move or rewrite functions developed at your site since FOCUS must switch to non-zIIP mode for each call to such routines. You may be able to use one of the following possible solutions:
  - Move the routines from DEFINEs to COMPUTEs to reduce the number of times they are referenced. This tactic must be applied carefully, and only when report results would not change.
  - Rewrite the routines using DEFINE FUNCTION, which executes on the zIIP processor.
  - Confine the routine to a pre-step run with ZIIP=OFF which collects its calculated results, then use those calculations in the next step with ZIIP=ON.

Obtaining APF Authorization for All Load Libraries

Authorization in batch is established by two trigger actions:

- The FOCUS module must have the AC/1 authorization attribute linked into the module. The FOCUS module is shipped with this attribute linked.
- Your system support staff must APF authorize the library containing the FOCUS module as well as all load libraries allocated in your JCL or CLIST. The FOCUS load library FOCLIB.LOAD and the FOCUS function library FUSELIB.LOAD are two examples. If your site has its own user-written functions, any load libraries in which these functions reside must also be authorized.

  If you have authorized the libraries properly, have a personalized FLICENSE file that licenses the zIIP feature, and set ZIIP=ON, the zIIP feature will be activated. If the authorizations are not correct, you will get the following error message:

  ```
  (FOC32577) ZIIP WILL NOT BE ACCESSED. NOT APF AUTHORIZED.
  ```

If you have been running in an authorized state for a while without problems and you suddenly get the 32577 message, something in your environment has changed. You have called a new module that was not called in prior runs, and this module is not authorized. See *Troubleshooting Authorization Errors* on page 171 for a technique you can use to find the module that is causing the problem.

Note: Authorization in TSO requires an additional trigger action. This step is described in *Obtaining AUTHCMD and AUTHPGM Authority for TSO Processing* on page 172.

All libraries that contain load modules must be APF authorized in SYS1.PARMLIB in the PROGxx member.

Example: Establishing an Authorized Environment for Using the zIIP Feature

Use the following sample of SYS1.PARMLIB(PROGxx) entries as a guide for obtaining APF authorization for your load libraries:

```
APF FORMAT(DYNAMIC)
 /* ============================================================== */
 /*    FOR APF AUTH LIBRARIES                                      */
 /* ============================================================== */
APF ADD DSNAME(hlq_prod.FOCLIB.LOAD)  VOLUME(volser)
APF ADD DSNAME(hlq_prod.FUSELIB.LOAD) VOLUME(volser)
```

where:

hlq_prod

Is the high-level qualifier for your FOCUS production libraries.

volser

Is the volume serial number of the disk pack containing the library.

Reference: Troubleshooting Authorization Errors

In a typical environment, a site will use its production FOCUS CLIST or JCL to validate the authorization process. Production CLISTs and JCL often allocate a large number of load libraries. This can make it difficult to tell which load libraries are not correctly authorized. Therefore, a good troubleshooting technique if you get the 32577 authorization message is to create a CLIST or JCL procedure that contains only the minimum number of libraries needed to run FOCUS. When that procedure works correctly, you can add other libraries one at a time until you get the 32577 message again. This would indicate that the last library you added to the CLIST or JCL was not authorized.

Note that in order to execute FOCUS, you must allocate the FOCLIB.LOAD, FUSELIB.LOAD, MASTER.DATA, FOCEXEC.DATA, and ERRORS.DATA libraries. You must issue the call to FOCUS from the authorized FOCLIB.LOAD library.

Obtaining AUTHCMD and AUTHPGM Authority for TSO Processing

If you run FOCUS under TSO, you must still APF authorize all load libraries that will be called by the FOCUS module, and the FOCUS module must be linked with the AC/1 attribute.

In addition, your system support staff must authorize the FOCUS program under TSO. To authorize the FOCUS program in TSO, have your system programmer update the IKJTSOxx member of the SYS1.PARMLIB for TSO Interactive (IKJTSO) and TSOBATCH (IKJEFT) programs of your site. They will have to authorize the command and program sections.

IBM restricts recursive calls to ISPF when you are operating in an authorized state. Therefore, if you execute FOCUS from ISPF option 6, you will not be able to issue the TSO ISPF or IEDIT commands from within FOCUS.

CALLs to a library are supported in both TSO and TSOBATCH environments.

RUNs to a library are supported in both TSO and TSOBATCH environments.

Example: Establishing an Authorized Environment for Using the zIIP Feature Under TSO

Use the following sample as a guide for obtaining AUTHCMD and AUTHPGM authority under TSO:

```
/* ************************************************************ */
/*                    SYS1.PARMLIB(IKJTSOxx)                    */
/* ************************************************************ */
AUTHCMD NAMES(          /* AUTHORIZED COMMANDS      */ +
   FOCUS                /* FOCUS                    */ +

AUTHPGM NAMES(          /* AUTHORIZED PROGRAMS      */ +
   FOCUS                /* FOCUS                    */ +
```

Activating a zIIP Environment or Projecting zIIP Usage

The last step in zIIP enablement is to activate the use of the zIIP processor in FOCUS. zIIP enablement is activated by the SET ZIIP command.

The SET ZIIP command has three options:

- **OFF.** This setting prevents FOCUS from offloading its processing to a zIIP.
- **ON.** This setting causes FOCUS to offload processing to a zIIP engine if you have a zIIP processor and the environment is properly authorized.

- **ON/SIMMAXZIIP.** This setting enables you to project zIIP processing in two different environments:
  - **You do not have a zIIP processor.** Using this setting along with the PROJECTCPU parameter, you can project how much FOCUS workload would have been offloaded to a zIIP.
  - **You do have a zIIP processor.** Using this setting you can project how much advantage you would achieve by offloading 100% of eligible FOCUS processing to the zIIP.

Syntax: **How to Activate the zIIP Enablement Feature**

You can issue the SET ZIIP command in FOCPARM or in a batch job stream.

```
SET ZIIP={ON[/SIMMAXZIIP]|OFF}
```

where:

`ON`

Configures FOCUS to offload processing to the zIIP engine.

This setting:

- Determines if the zIIP processor is accessible to the LPAR in which a job is running.
- Determines if the FOCUS (Batch or TSO) environment has been properly authorized to run a zIIP workload.

Note: If FOCUS determines that the zIIP processor is not accessible or that the environment has not been authorized correctly, it issues a message describing the reason and continues in ZIIP=OFF mode, which forwards all subsequent work to the Central Processor.

`ON/SIMMAXZIIP`

Configures FOCUS to either:

- Project what the zIIP usage would be if FOCUS could offload processing to a zIIP, when FOCUS is operating in an LPAR without a zIIP. This requires that the PROJECTCPU parameter be set to YES.

  The SYS1.PARMLIB member IEAOPTxx contains the PROJECTCPU statement. Activating the PROJECTCPU parameter projects zIIP consumption when a zIIP processor is not yet defined to the LPAR. SMF type 30 records will show the potential calculated zIIP time, so that you can accurately project zIIP usage. This enables you to evaluate the effect of configuring a zIIP processor to be available for FOCUS usage. Your site's Systems Programmer will have access to this data. Use this option for simulation purposes only.

Since the zIIP engine actually is not present, all zIIP-eligible workload will be diverted to the Central Processor. Thus all of that CPU utilization will be recorded in a FOCUS variable called &FOCZIIPONCP. This is the amount of workload that would have run on the zIIP engine, and would have appeared in &FOCZIIPCPU, had the zIIP been present and accessible to FOCUS work. This information is also recorded in the log file of the FOCUS Utilization Reporting utility (FOCLOG) as well as in IBM SMF type 30 records.

To use this option, insert the following parameter in SYS1.PARMLIB for your LPAR, and also issue the SET ZIIP=ON/SIMMAXZIIP command:

```
PROJECTCPU=YES
```

This setting:

- Determines if the PROJECTCPU=YES command has been set in the LPAR.
- Determines if the FOCUS (Batch or TSO) environment has been properly authorized to run a zIIP workload.

- Project zIIP utilization if 100% of eligible FOCUS processing could be offloaded to the zIIP, when FOCUS is running in an LPAR with a zIIP. This lets you determine what you would gain by configuring Workload Manager to give FOCUS a bigger share of zIIP processing.

  IBM Workload Manager (WLM) prioritizes workloads among the Central Processors and zIIP processors at your site based on a complex set of goals and rules established by the system administrator. These rules apply to all workloads from all sources, not just FOCUS. These goals combine to influence the decision to direct FOCUS requests to the zIIP engine at any particular moment.

  Utilizing this setting with a zIIP present can help you determine how much advantage you would get if FOCUS had more of a share of the zIIP processor. To see the difference in actual and projected zIIP usage, run the same job with SET ZIIP=ON and then with SET ZIIP=ON/SIMMAXZIIP and compare the results. For more information about evaluating zIIP usage, see *Evaluating zIIP Usage* on page 178.

  This setting:

  - Determines if the zIIP processor is accessible to the LPAR in which a job is running.
  - Determines if the FOCUS (Batch or TSO) environment has been properly authorized to run a zIIP workload.

Note: If FOCUS determines that the environment has not been authorized correctly, it issues a message describing the reason and continues in ZIIP=OFF mode, which forwards all subsequent work to the Central Processor.

OFF

Configures FOCUS not to offload processing to the zIIP engine. OFF is the default value.

Information Builders Note: Turn off zIIP enablement only when you know for sure that a job will not gain any advantage from using the zIIP processor or if the system operator or administrator requires that you turn it off.

Example: Setting the PROJECTCPU Parameter in SYS1.PARMLIB Member IEAOPTxx

Use the following sample as a guide for setting the PROJECTCPU parameter in SYS1.PARMLIB(IEAOPTxx):

```
/* ************************************************************ */
/*                    SYS1.PARMLIB(IEAOPTxx)                    */
/* ************************************************************ */
PROJECTCPU=YES
```

Establishing a Non-authorized Environment After Enabling the zIIP Feature

Your site may not allow certain processes to run in an authorized state. Therefore, there may be times when you need to run FOCUS in an unauthorized state. Some examples of such processes are ENDEAVOR or recursive calls to ISPF.

If you run such processes, you need to have both the authorized and non-authorized FOCUS environments available for use at your site. The authorized FOCUS environment will use the zIIP processor. The non-authorized FOCUS environment will enable you to run processes such as ENDEAVOR, but it will not provide the benefit of offloading FOCUS processing to the zIIP engine.

Example: Establishing a non-Authorized Environment While Using the zIIP Feature

There are two methods you can use to bring up a non-authorized version of FOCUS if you have FOCUS configured to utilize the zIIP, but run into a situation where a program called by FOCUS requires a non-authorized state in order to execute.

Method 1: Calling FOCUS From a Different Library

This method works only if FOCUS processing is run only in batch, not under TSO. For this method, make a copy of the production FOCLIB.LOAD library, call it FOCLIBNA.LOAD, and *do not* APF authorize this load library. Replace FOCLIB.LOAD with the FOCLIBNA.LOAD library in your batch FOCUS JCL.

Method 2: Calling FOCUS By a Different Name

The second method works in both FOCUS batch and TSO environments. For this method, you need to create a new FOCUS module called FOCUSNA which does not have the AC/1 authorization attribute linked. You can place this module either in the production FOCLIB.LOAD library or in a new load library (for example, FOCLIBNA.LOAD). Change your CALL statement to call FOCUSNA instead of FOCUS from the relevant library.

The JCL to create the FOCUSNA module is distributed as member FOCUSNA in the FOCCTL.DATA library:

1. Copy the FOCUSNA member to your FOCUS production JCL library.
2. Make a copy of your FOCLIB.LOAD library and call it FOCLIBNA.LOAD.
3. Instructions for editing the JCL for your site are included as comments in the FOCUSNA member. Make the recommended edits to the FOCUSNA JCL.
4. Execute the FOCUSNA JCL.
5. Check the FOCLIBNA.LOAD library for the presence of the new FOCUSNA module and EDASAFNA alias.

   Please note that the FOCUSNA module will have an authorization attribute of AC/0, which means that the module was not linked with authorization.

The following is a sample CLIST for calling the FOCUSNA program from a new load library named FOCLIBNA.LOAD in TSO:

```
CONTROL MSG LIST
WRITE ************************************************************
WRITE
WRITE    Sample FOCUS CLIST
WRITE
WRITE ************************************************************
WRITE
FREE FI(USERLIB FOCLIB ERRORS MASTER FOCEXEC)
ALLOC F(USERLIB)  DA('hlq_test.FOCLIBNA.LOAD' -
                    'hlq_prod.FUSELIB.LOAD') SHR REU
ALLOC F(MASTER)   DA('hlq_prod.MASTER.DATA') SHR REU
ALLOC F(FOCEXEC)  DA('hlq_prod.FOCEXEC.DATA') SHR REU
CALL 'hlq_test.FOCLIBNA.LOAD(FOCUSNA)'
```

where:

hlq_prod

Is the high-level qualifier for your FOCUS production libraries.

hlq_test

Is the high-level qualifier for the new FOCLIBNA.LOAD library that contains the FOCUSNA module. After testing this method, you can move the FOCLIBNA.LOAD library to the production environment along with your other FOCUS production libraries, by copying it to the same high-level qualifier as your FOCLIB.LOAD library.

How FOCUS Takes Advantage of the zIIP Processor

FOCUS diverts eligible workload to the zIIP engine by switching from TCB (Task Control Block) mode for workloads that can run only on a Central Processor to SRB (Service Request Block) mode for execution of enabled workloads on the zIIP engine.

Types of FOCUS processing that are offloaded to the zIIP engine include:

- Computations.
- Aggregation.
- Screening.
- Report formatting and styling.
- Transaction Processing.

The FOCUS zIIP Monitor detects situations in which the overhead cost of zIIP usage is exceeding the CPU benefits gained. When this threshold is reached, FOCUS may decide to suspend use of the zIIP for the duration of that command. It then resets to make the zIIP processor accessible to the next command.

TABLE, MATCH, and MORE requests may suspend and resume more than once as they progress through logical phases of execution.

In every case, FOCUS attempts to optimize the use of the zIIP and minimize chargeable CPU costs.

Applications that perform significant database I/O, high-volume sorting, or the use of third party tools or user functions during processing require switching out of SRB (zIIP) mode into TCB (non-zIIP) mode to communicate, and then back again to continue processing. Although each switch is miniscule, the cumulative effect can absorb measurable amounts of CPU time on both the zIIP engine and the Central Processor.

In order to diminish this effect, FOCUS buffers the collection of records passed to the system sort utility and some adapters rather than passing one record at a time, thus reducing the number of switches between TCB and SRB modes.

Enabling zIIP automatically turns off the external sort (SET EXTSORT=OFF), as this may provide better performance. You can turn the external sort back on after enabling zIIP, if required.

These third party products may themselves be zIIP enabled and may offload some or all of their processing to the zIIP engine independent of FOCUS. FOCUS always calls these products from the Central Processor because it cannot know whether they will perform any processing that is prohibited on the zIIP.

Even though zIIP usage occurs more frequently on non-optimized requests to a relational data source, optimized requests are still inherently more efficient and, therefore, may incur less CPU time. Being zIIP enabled, DB2 may also take advantage of the zIIP processor for FOCUS requests based on the local configuration of DB2 relative to FOCUS.

Requests against some types of data sources whose I/O can be buffered gain a lot of advantage from zIIP enablement. Data sources that gain the most benefit from zIIP processing due to buffered I/O include:

- Blocked flat files.
- FOCUS.
- XFOCUS.
- VSAM.
- DB2.

Evaluating zIIP Usage

In order to evaluate FOCUS zIIP processing in a session, you can query three Dialogue Manager variables that accumulate statistics about FOCUS processing:

- &FOCCPU accumulates the time spent on a Central Processor. This is an existing variable that precedes zIIP enablement.
- &FOCZIIPCPU accumulates the time actually spent on the zIIP processor (in SRB mode). This is the normalized CPU value using the same scale as &FOCCPU.
- &FOCZIIPONCP accumulates the time that processing could have been offloaded to the zIIP processor but was diverted to the Central Processor by the system.

  Note:

  - &FOCCPU includes the value of &FOCZIIPONCP.
  - The sum of &FOCZIIPCPU and &FOCCPU represents the total CPU utilized by the job.

- If you set ZIIP=OFF, the zIIP variables do not accumulate further but are not reset to zero. If you later set ZIIP=ON, they resume accumulating statistics.

The FOCLOG and SiteAnalyzer products that monitor FOCUS usage also capture zIIP statistics for a session.

Calling FOCUS Under TSO

You can allocate the file names required by FOCUS:

- In the logon procedure.
- Using normal batch JCL.
- Using TSO CLISTs that you execute after the logon, before FOCUS is entered.
- From within FOCUS, prior to their first use, with DYNAM ALLOCATE or TSO ALLOCATE commands. Information Builders recommends using DYNAM as its performance is superior to passing each command to TSO.

A common practice is to allocate all commonly used files in the logon procedure, and the remainder from a TSO CLIST.

Batch Operation

You can process FOCUS jobs in z/OS batch mode. The z/OS operator can initiate the jobs directly, or you can enter them in z/OS using the TSO SUBMIT or DYNAM SUBMIT command (see *DYNAM Command* on page 200 for DYNAM commands).

Note: When you design FOCUS applications for execution both online and in batch, it is important to remember that in z/OS batch mode, FOCUS ignores embedded TSO commands in FOCUS procedures. You must, therefore, supply DD statements to replace any TSO ALLOCATE commands or use the DYNAM ALLOCATE command.

Example: Processing FOCUS Jobs in z/OS Batch Mode

The following is a sample job stream of z/OS batch JCL:

```
//FOCUS EXEC PGM=FOCUS
//STEPLIB  DD DSN=FOCUS.FOCLIB.LOAD,DISP=SHR
//ERRORS   DD DSN=FOCUS.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=A
//OFFLINE  DD SYSOUT=A
//MASTER   DD DSN=MASTER.DATA,DISP=SHR
//FOCEXEC  DD DSN=FOCEXEC.DATA,DISP=SHR
//* FOCUS FILE CAR
//CAR      DD DSN=CAR.FOCUS,DISP=SHR
//SYSIN    DD *
 .
 . FOCUS commands
 .
FIN
//*
```

Each time you enter FOCUS, it automatically opens and executes the procedure PROFILE if it is present before opening SYSIN. If you operate FOCUS in batch mode, you should include a FIN statement in the PROFILE or in the SYSIN jobstream. Otherwise, FOCUS will terminate with completion code 8.

Note that outside of a request or FOCEXEC, you can insert a comment line in SYSIN by starting the line with an asterisk (*).

To use an alternative PROFILE member in place of the usual one, specify the following values for the PARM parameter:

```
//FOCUS EXEC PGM=FOCUS,PARM='NOPROF'
```

or

```
//FOCUS EXEC PGM=FOCUS,PARM='PROFILE member'
```

where:

`NOPROF`

Is an optional parameter. Enables you to bypass or ignore your usual PROFILE member of the FOCEXEC data set when you enter your FOCUS session.

`member`

Is an optional parameter. Names an alternative PROFILE to execute instead of the usual PROFILE member.

Direct Entry

The FOCUS load module that controls the FOCUS environment resides in the partitioned data set FOCLIB.LOAD, member name FOCUS. The data set name may vary due to local data set naming conventions.

You can enter FOCUS either by having a logon procedure in which the load library is allocated to DDNAME STEPLIB, by issuing the CALL command directly, or through a CLIST.

Example: Entering FOCUS

A logon procedure is a convenient way of allocating frequently used data sets.

The following is a sample TSO logon procedure:

```
//TSOLOGON EXEC PGM=IKJEFT01,DYNAMNBR=50
//*
//STEPLIB DD DSN=FOCUS.FOCLIB.LOAD,DISP=SHR
//ERRORS  DD DSN=FOCUS.ERRORS.DATA,DISP=SHR
//OFFLINE DD SYSOUT=A
//* USUAL TSO LOGON DATASETS FOLLOW
```

If you have a logon procedure with FOCLIB.LOAD allocated to DDNAME STEPLIB, to invoke FOCUS, simply issue:

```
FOCUS
```

This places you in the FOCUS environment.

To use an alternative PROFILE member in place of the usual one, specify:

```
FOCUS {NOPROF|'PROFILE member'}
```

where:

`NOPROF`

Is an optional parameter. Enables you to bypass or ignore your usual PROFILE member of the FOCEXEC data set when you enter your FOCUS session.

`member`

Is an optional parameter. Names an alternative PROFILE to execute instead of the usual PROFILE member.

You can also enter the CALL command directly:

```
CALL 'FOCUS.FOCLIB.LOAD(FOCUS)'
```

Example: Using CLISTs

The following is a typical FOCUS CLIST procedure that allocates files:

```
ALLOC F(MASTER)   DA(MASTER.DATA)
ALLOC F(FOCEXEC)  DA(FOCEXEC.DATA)
ALLOC F(CAR)      DA(CAR.FOCUS) SHR
ALLOC F(HOLD1)    SP(5,5) TRACKS

CALL 'FOCUS.FOCLIB.LOAD(FOCUS)'
```

Note: SYSIN and SYSPRINT do not have to be allocated and will default to the terminal.

The following is an example of a more comprehensive CLIST used in a typical FOCUS production environment:

```
PROC  0
/*                                                              */
CONTROL NOMSG
FREE  F(MASTER CCARS CAR FOCEXEC ERRORS USERLIB FMU)
FREE  F(SYSIN SYSPRINT OFFLINE)
FREE  F(ISHFALOC PROFILE)
FREE  F(HOLD HOLDMAST SAVE REBUILD FOCSML FOCUS FOCSTACK)
FREE  F(FOCSORT HOLDACC TRF)
/*                                                              */
CONTROL MSG
/*                                                              */
ALLOC F(PROFILE)   DA('user.FOCEXEC.DATA(PROFILE)')       SHR REUS
ALLOC F(FOCEXEC)   DA('user.FOCEXEC.DATA'              -
                      'prefix.FOCEXEC.DATA')              SHR REUS
ALLOC F(MASTER)    DA('user.MASTER.DATA'               -
                      'prefix.MASTER.DATA')               SHR REUS
/*                                                              */
ALLOC F(FMU)       DA('prefix.FMU.DATA')                  SHR REUS
ALLOC F(TRF)       DA('user.TRF.DATA')                    SHR REUS
/*                                                              */
ALLOC F(CCARS)     DA('prefix.MASTER.DATA(CCARS)')        SHR REUS
ALLOC F(CAR)       DA('user.CAR.FOCUS')                   SHR REUS
/*                                                              */
ALLOC F(ERRORS)    DA('prefix.ERRORS.DATA')               SHR REUS
ALLOC F(USERLIB)   DA('prefix.FUSELIB.LOAD')              SHR REUS
/*                                                              */
ALLOC F(OFFLINE)   SYSOUT(A)
ALLOC F(SYSIN)     DA(*)
ALLOC F(SYSPRINT)  DA(*)
/*                                                              */
CALL 'prefix.FOCLIB.LOAD(FOCUS)'
/*                                                              */
```

```
CONTROL NOMSG
FREE  F(MASTER CCARS CAR FOCEXEC ERRORS USERLIB FMU)
FREE  F(SYSIN SYSPRINT OFFLINE)
FREE  F(ISHFALOC PROFILE)
FREE  F(HOLD HOLDMAST SAVE REBUILD FOCSML FOCUS FOCSTACK)
FREE  F(FOCSORT HOLDACC TRF)
```

where:

user

Is the high-level qualifier for a user's version of a data set.

prefix

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

Note: The allocation for DDNAME CCARS is needed for running the CARTEST FOCEXEC.

FOCUS Facilities Under TSO

FOCUS facilities under TSO include:

- FIDEL, for describing full-screen data entry forms.
- TED, an optional full-screen editor for creating and editing text files.
- IEDIT, a facility for invoking your system editor from FOCUS.
- GRAPH, a command used to generate graphic displays such as histograms, bar charts, and connected point plots.
- REBUILD, a utility that enables you to make structural changes to an existing FOCUS data source.

Using FIDEL

The actual terminal control program is loaded dynamically, according to the program name you specify in the DATA VIA subcommand of MODIFY.

```
DATA VIA FIDEL
END
```

A DATA statement is not required in a MODIFY which uses CRTFORM if the procedure will be executed on a 3270-type terminal. FOCUS automatically loads FIDEL if the MODIFY procedure uses CRTFORM.

TED Editor

You can edit a file with TED using the TED command. You can also edit a file with your system editor using the IEDIT facility. For information about IEDIT, see *Invoking the ISPF Editor on z/OS* on page 83.

Syntax: How to Enter TED

```
TED ddname
```

or

```
TED 'prefix.qualifier1.qualifier2...(member)'
```

or

```
TED name
```

where:

prefix

Is a prefix other than the one you are working under. If you are working within your own prefix, you do not have to specify the prefix or the single quotation marks around the data set name.

qualifier

Is the name of the file within the prefix. Qualifiers are separated with periods. You can have up to 44 characters in a data set name.

(*member*)

Is necessary only when you are using a partitioned data set. For sequential data sets, a member name (enclosed in parentheses) is not used.

name

Is either the DDNAME of an allocated sequential data set or the member of a partitioned data set allocated to DDNAME FOCEXEC. Specify only from the FOCUS command line.

FOCUS searches for the member of a partitioned data set allocated to DDNAME FOCEXEC first. If a partitioned data set does not exist, FOCUS continues and searches for a sequential data set allocated to DDNAME *name*.

Example: Entering TED

You can specify a valid data set name or DDNAME with a member name if the file is a partitioned data set. Note the following examples:

`TED`	Edits the last executed FOCEXEC.
`TED TEST`	Edits the data set allocated to DDNAME TEST or the member name in a partitioned data set allocated to DDNAME FOCEXEC.
`TED FOCEXEC(A)`	Edits member A of the data set allocated to DDNAME FOCEXEC.
`TED X.DATA(REPT)`	Edits member REPT of the data set *prefix*.X.DATA.
`TED 'USER1.X.DATA'`	Edits the data set USER1.X.DATA.

Syntax: **How to Issue TSO Commands From TED**

`{MVS|TSO}` *`command`*

where:

`MVS|TSO`

Specifies the operating system. The MVS prefix may be substituted for the TSO prefix.

`command`

Is a TSO command

Note:

- In z/OS, TED cannot edit uncataloged data sets.
- In z/OS, the execution of a fully qualified data set name as a FOCEXEC does not allow the TED command to be issued without that full name following the command.
- TSO commands may be issued on the command line in TED, but not within Screen Painter.
- The maximum record length supported by TED in TSO is 32760 bytes.
- TED may not edit files containing unprintable characters.
- In z/OS, the ISPF statistics for date, time, version, size, and user ID are automatically updated when the member of a partitioned data set is stored using the TED FILE or SAVE command.

Syntax: How to Submit a Job From TED

You can submit a job from TED by using the following syntax to submit the current file to z/OS:

```
SUBmit
```

Example: Creating a TED Profile

On entry to TED, a search is made for a profile as member TEDPROF of the data set allocated to DDNAME FOCEXEC. If found, it is processed by TED before the first screen appears. The profile may issue most TED commands that do not involve any file manipulation (for instance such commands as SPV and FILE may not be issued in the profile). The most common use for a profile is to establish a prefix area and perhaps a scale and PF key assignments. A typical profile might be:

```
1. NUM ON
2. SCALE ON
3. PF 6 FILE
```

where:

1. Sets the prefix area on and displays line numbers.
2. Displays a scale on line 2 of the screen.
3. Redefines the PF6 key as FILE.

National Language Support

FOCUS is designed with consideration for National Language Support (NLS). FOCUS stores data as entered and retrieves the data based on the current code page mapping. FOCUS can be configured for local language messages and keyboards. For installation instructions, refer to the *z/OS Installation Guide*. Your Information Builders' representative can provide a list of available language choices.

The ? LANG command can be used to determine the current language setting and parameters pertaining to language, such as continental decimal notation.

Syntax: How to Determine Current Language Settings

```
? LANG
```

FOCUS supports languages that require two bytes of storage per character, such as Kanji. For more information, see the *Describing Data* manual.

TSO and FOCUS Interaction

You can execute TSO commands from within FOCUS. These commands can be used in conjunction with FOCUS to create and maintain applications. You can also go directly to ISPF edit screens from within the FOCUS command environment. This means that you can start a FOCUS session and then issue TSO and ISPF commands to allocate files, create, and edit FOCUS procedures and Master Files, all directly from with FOCUS.

Additionally, you can issue command interrupts to halt processing or suppress output.

Issuing TSO Commands From Within FOCUS

You can issue almost all of the standard TSO commands from within the FOCUS environment. The exceptions are EXEC, TIME, and TSO commands that load programs, such as CALL and COMPILE.

This facility applies only to interactive FOCUS sessions. FOCUS ignores TSO commands when running in batch mode, and does not generate error diagnostics. You can still run unaltered procedures in batch mode and TSO, but you must add supplementary JCL statements in the batch mode version to replace any TSO ALLOC statements embedded in a FOCUS procedure.

Syntax: How to Issue a TSO Command From Within FOCUS

```
{TSO|MVS} system command parameters
```

Note: The MVS prefix may be substituted for the TSO prefix.

The initial keyword, TSO or MVS, tells FOCUS how to interpret what follows and appears only once on the first line of the command. Otherwise, the syntax of the TSO command and accompanying parameters is the same whether you are inside or outside of FOCUS. For example:

```
TSO ALLOCATE F(SAVE1) SPACE(5 5) TRACKS
```

TSO commands are documented in the *IBM TSO User's Guide*. Installation written commands should be avoided or thoroughly tested, as they may or may not execute properly within FOCUS.

Example: Executing a CLIST From Within FOCUS

You can execute TSO Command Lists (CLISTs) from within FOCUS by invoking the services of ISPF. To execute a CLIST from within FOCUS, all of the files needed to run ISPF must be allocated and the CLIST to be invoked must exist as a member of a partitioned data set allocated to DDNAME SYSPROC. For example:

```
ALLOC FILE(SYSPROC) DA('WIBMBP.M.CLIST') SHR
```

There are two ways of running the CLIST, depending on whether FOCUS was entered from TSO or from within ISPF.

- If FOCUS was entered from the TSO command level, the command

  ```
  TSO ISPSTART CMD(memname)
  ```

  executes CLIST *memname* from the library allocated to SYSPROC (WIBMBP.M.CLIST in this case). The ISPSTART command could be incorporated in a LET statement such as

  ```
  LET GOCLIST = TSO ISPSTART CMD (< >)
  ```

  and may be issued at the FOCUS command level by specifying:

  ```
  GOCLIST memname
  ```

- If FOCUS was entered from within ISPF, a FOCEXEC should be created to invoke the ISPLINK processor. The FOCEXEC is needed to pass the correct parameters to the ISPLINK utility. The parameter COMLEN, which specifies the length of the command, must be passed as a binary integer. For example:

  ```
  -SET &COMAND  = 'CMD('|| &MEMNAME.Enter name of CLIST. || ')';
  -SET &COMLEN1 = HEXBYT(0,'A1');
  -SET &COMLEN2 = HEXBYT(&COMAND.LENGTH,'A1');
  -SET &COMLEN  = &COMLEN1 || &COMLEN1 || &COMLEN1 || &COMLEN2;
  -TSO RUN ISPLINK,SELECT,&COMLEN,&COMAND
  ```

Note:

- Some commands placed in a CLIST that is invoked from within FOCUS may give unpredictable results, particularly when FOCUS has been invoked from within ISPF.
- For detailed documentation on ISPSTART or ISPLINK, consult the *System Productivity Facility (ISPF), Dialogue Management Services*, or contact your system support group.

Using TSO Commands in FOCUS Applications

Four TSO commands, COPY, LIST, LISTDS, and RENAME, are frequently used within the FOCUS environment to create and maintain FOCUS Master Files and FOCEXECs. TSO COPY and TSO LIST are program product commands, available only if you have exercised those options. From within FOCUS, you can reference data sets in the command segments either by actual data set names (as in normal TSO syntax) or by DDNAMEs. Obviously, in most instances, the short DDNAME syntax will appeal simply for ease of entry. The FOCUS syntax for the DDNAME versions of the four TSO commands follows:

```
TSO COPY   ddname datasetname
TSO LIST   ddname
TSO LISTDS ddname
TSO RENAME ddname datasetname
```

FOCUS recognizes that you are using the DDNAME syntax by the absence of embedded periods (.) in the name field of the argument. When FOCUS receives one of these commands, it replaces the DDNAME portion of the argument with the fully qualified data set name for that DDNAME. FOCUS then reprints the translated command on your terminal as an informational message before sending it to TSO for execution.

For example, if you want to rename the file, GM.FOCUS (allocated as DDNAME CAR), to data set name CHEVY.FOCUS, you enter:

```
TSO RENAME CAR CHEVY.FOCUS
```

FOCUS responds with an informational message:

```
RENAME 'ID.GM.FOCUS' CHEVY.FOCUS
```

TSO then receives and executes the command.

Note:

- If you want to save a temporary data set, you cannot use RENAME; you must use COPY, since RENAME does not catalog a temporary data set.
- In TSO commands that use more than one data set, such as copy, you can only replace the first data set name with a DDNAME.

If you want to suppress the printing of the informational messages, you can do so by issuing the FOCUS command SET MESSAGE=OFF.

The argument in any of the TSO commands mentioned in the previous paragraphs can also refer to a member in a partitioned data set by including the member name within parentheses after the DDNAME under which the data set was allocated. For example, the command TSO LIST INDATA(CAR), calls the TSO command LIST to display member CAR in the data set allocated as INDATA. The DDNAME can be any DDNAME, including significant FOCUS DDNAMEs such as MASTER or FOCEXEC.

You can use the FOCUS DDNAME TSO command syntax to refer to concatenated data sets in any case to make sure that the reference cannot be interpreted unambiguously. FOCUS evaluates the combination of DDNAMEs, member names, data set organizations, and concatenations, and then diagnoses the command. If the command request is unambiguous, it is executed. If the command could have ambiguous results, it is not executed and a message is returned.

For example, if you wish to display a member named ACCOUNTS that is part of the only data set allocated to DDNAME FOCEXEC, you enter:

```
TSO LIST FOCEXEC(ACCOUNTS)
```

FOCUS displays the member, ACCOUNTS, in the data set allocated to FOCEXEC.

If, in the same example, there were several partitioned data sets allocated to DDNAME FOCEXEC and TSO EDIT was called, FOCUS would diagnose the potential problem and would not process the edit request because it would not know where to put the output. Instead, FOCUS would return an error message to the terminal.

FOCUS Command Interrupt Levels

When you are in the FOCUS command environment, you can issue an external interrupt to stop execution of some commands (MODIFY, FSCAN, TABLE, TABLEF, MATCH, and GRAPH). This results in an orderly closing of the FOCUS data source and/or suppressing the output. To issue an interrupt, press one of the following function keys:

Terminal	Function Key
IBM 327X	PA1 or ATTN
IBM 2714	ATTN
ASCII	BREAK
ASCII (full duplex)	ESC

FOCUS signals receipt of the interrupt by the message

```
FOCUS INTERRUPTED..ENTER KX,KT,RT, FX OR ?
```

and waits for the user's reply. The effect of each reply is as follows:

KX	Kill execution, stay in FOCUS.
KT	Kill typing until next terminal read.
RT	Resume typing.

FX	Kill execution, exit FOCUS.
?	Display current run-time statistics.
Other	Ignored, execution resumes.

Reference: **Kill Execution: KX**

This reply stands for "Kill Execution" and remains in FOCUS. It can be used with the GRAPH, MATCH, MODIFY, TABLE, TABLEF, and FSCAN commands. It tells FOCUS to halt execution and return you to the FOCUS command level outside all FOCEXEC procedures (that is, to the next command from SYSIN, which is generally the terminal). The interrupted command may terminate normally or it may be cut short. The FOCUS data source is closed in an orderly manner. The commands are terminated as follows.

MODIFY	At end of current transaction.
SCAN	At end of current subcommand.
TABLE, TABLEF, MATCH and GRAPH	At next data access or generation of an output line.
All others	Ignore the KX reply and continue to completion.

Reference: **Kill Typing: KT**

This reply stands for "Kill Typing" and can be used with GRAPH, MODIFY, FSCAN, TABLE, and TABLEF. It tells FOCUS to suppress output of the command but to allow the command to continue to completion. After you enter KT, nothing more will appear on the screen until the command finishes, when FOCUS will prompt you for the next command.

The KT feature is useful when FOCUS is producing a report of unexpected length. Suppressing output eliminates terminal I/O and speeds up processing. You can save the output in a file with the SAVE and HOLD commands, or you can display the output from the beginning with the RETYPE and REPLOT commands.

Reference: Resume Typing: RT

This reply stands for "Resume Typing" and is used after entering the KT subcommand in response to a previous interrupt. After you enter KT, nothing will appear on the screen until the command is finished executing. To have FOCUS resume display of the output, press the interrupt key and enter RT. FOCUS displays output with the first output record it produces after this latest interrupt.

The RT interrupt reply is useful for discovering how far FOCUS has gone in producing output. If you want to suppress output again, press the interrupt key and enter KT.

Reference: Kill Execution: FX

This reply stands for "Kill Execution" and exit FOCUS. It can be used with the GRAPH, MATCH, MODIFY, TABLE, TABLEF, and FSCAN commands. It tells FOCUS to halt execution and return you to the TSO session. The interrupted command may terminate normally, or it may be cut short. The FOCUS data source is closed in an orderly manner. The commands are terminated as follows:

`MODIFY`	At end of current transaction.
`SCAN`	At end of current subcommand.
`TABLE, TABLEF, MATCH and GRAPH`	At next data access or generation of an output line.
`All others`	Ignore the FX reply and continue to completion.

Reference: Display Statistics: ?

This reply can be used with the commands TABLE, TABLEF, GRAPH, MATCH, and MODIFY. It displays statistics on reports of record modifications that FOCUS has processed. Afterwards, FOCUS displays the output from the point where it was interrupted. The statistics are: the number of records in the report or the records modified by the MODIFY or FSCAN command, the number of lines in the report, and the number of I/O operations FOCUS performed to read or modify the data source.

Reference: Interrupting TSO Commands

When you are executing a TSO command from within the FOCUS environment, you can issue an interrupt to cancel execution of the command by pressing down once on the appropriate function key. FOCUS signals receipt of the interrupt with the message:

```
FOCUS INTERRUPTED..ENTER KX,KT,RT, FX OR ?
```

Enter KX. This will terminate execution of the TSO command and return you to the FOCUS environment. Any other response will have no effect and you will simply remain in suspended operation. You must hit interrupt again and make the correct KX response.

ISPF From FOCUS

Under TSO, you can enter ISPF edit screens directly, without going through the ISPF menus. To directly enter an edit screen from FOCUS, you must create a procedure that issues the EDIT DATASET command. This procedure must be a member of a partitioned data set that is concatenated to the allocation for DDNAME SYSPROC. The steps for creating such a procedure are as follows:

1. Create a data set (LRECL=80, RECFM=FB, BLKSIZE=1600) to contain your procedure.
2. Concatenate the data set to your allocation for DDNAME SYSPROC.
3. Create a member in this data set to contain your procedure. Give the member a name that reminds you of the fact that you use it to edit screens, but do not give it the name of an actual ISPF or TSO function. For example, you can call the member EDITIT.

   This member should contain the following procedure:

   ```
   PROC 1 ARG
   ISPEXEC EDIT DATASET(&ARG)
   ```

   When you call this procedure, it will take as an argument the fully qualified data set name of the member or sequential file you want to edit, and it will open an edit screen for that member or file.

In order to call your procedure and open an ISPF edit screen from FOCUS, you must enter FOCUS from the TSO Ready prompt, not from ISPF 6.

Syntax: How to Call a Procedure From Within FOCUS

```
TSO ISPSTART CMD(procname 'file_to_edit')
```

where:

procname

Is the name of the member that contains your procedure.

file_to_edit

Is the fully qualified name of the sequential file or data set member that you want to edit, enclosed in single quotation marks.

For example, if your procedure is called EDITIT and you want to edit member MYFILE in the data set named USER1.MASTER.DATA, issue the following command:

```
TSO ISPSTART CMD(EDITIT 'USER1.MASTER.DATA(MYFILE)')
```

ISPF From FOCUS From ISPF

The previous technique will only work if FOCUS has been entered directly from TSO command level. It will not work if FOCUS has been entered from option 6 of ISPF, since it calls ISPF. ISPF cannot be called when running under ISPF, either directly or indirectly.

ISPF Service Procedures do allow parts of ISPF to be called from within another program. By combining this facility with FOCUS' ability to dynamically call subroutines, the ISPF within ISPF limitation can be overcome. To accomplish this using FOCUS facilities and the ISPF facilities known as ISPLINK and ISPEXEC, follow these steps:

1. Prior to entry into ISPF, the FOCUS load library (FOCLIB.LOAD) must be in a system search library. This may be STEPLIB, LINKLIB, or some other standard system DDNAME. If not, the user (or systems person) can put it under the ISPF library name of ISPLLIB. Once done, ISPF can be entered through normal means.
2. From within ISPF, FOCUS may be entered from option screen 6 or some other screen. However, instead of a direct call to FOCUS, FOCUS should be entered by a call to the ISPF facility called ISPEXEC. This command cannot be entered directly, but it can be executed from within a CLIST. The syntax is:

   ```
   CONTROL NOMSG
   ISPEXEC SELECT PGM(FOCUS)
   ```

3. Once within FOCUS, the ISPF edit screens can be invoked by calling a second ISPF facility called ISPLINK. ISPLINK must be dynamically called using the -TSO RUN Dialogue Manager command. The first parameter of the call is the command (EDIT or BROWSE), and the second parameter is the data set name. If a partitioned data set is specified without a member, a member list screen is provided.

   A sample FOCEXEC with the necessary syntax follows:

   ```
   -TSO RUN ISPLINK, CONTROL, DISPLAY, REFRESH
   -TSO RUN ISPLINK, EDIT, FOCEXEC.DATA(&1.A8.FOCEXEC NAME.)
   TSO PROFILE PROMPT
   ```

   In the sample, the first -TSO RUN statement prevents screen erasure errors that may occur when entering ISPF after FOCUS performs a full screen I/O.

Note the TSO command after the call. This is required to ensure that when the FOCEXEC file is next accessed, the updated FOCEXECs are used. If your FOCEXEC file is not called FOCEXEC.DATA, the fully qualified data set name may be used. If not fully qualified, the TSO PREFIX is appended to the data set name.

ISPLINK does provide the facilities to call other parts of ISPF, though less directly. These can be provided upon request from Information Builders or through your systems support group.

Reviewing Attributes of Allocated Files

FOCUS enables you to check on existing file allocations within the interactive environment. The FOCUS command ? TSO DDNAME provides this facility through three optional command formats:

`? TSO DDNAME`	For listing allocated files.
`? TSO DDNAME` *ddname*	For listing file attributes.
`-? TSO DDNAME` *ddname*	For placing file attribute information into Dialogue Manager variables.

The first two formats are direct FOCUS commands that you can enter live at run time or place in a FOCEXEC. Both produce a listing on file SYSPRINT (ordinarily the TSO terminal) even if you issue the OFFLINE command. The third format can occur only in a FOCEXEC procedure and does not produce any visible output. Instead, the attributes of the queried DDNAME are returned as values for Dialogue Manager variables.

Note: The MVS prefix may be substituted for the TSO prefix.

Example: Listing the Currently Allocated Files

The ? TSO DDNAME command lists all the currently allocated files by DDNAME, shows the number of occurrences for each, and lists their corresponding data set names. It displays currently allocated files regardless of whether the allocations took place in the logon procedure or through TSO ALLOCATE and DYNAM ALLOCATE commands. Sample output follows:

```
>>? TSO DDNAME
DDNAME          OCCURRENCES     DSNAME
STEPLIB              2          TSO.TS.FOCUS.LOADLIB
                                INPDQ.PROC.PROCLIB
MASTER               1          INPDQ.MASTER.DATA
FOCEXEC              1          INPDQ.FOCEXEC.DATA
ERRORS               1          INPDQ.ERRORS.DATA
FOCSTACK             1          SYS88229.TO95525.RA000.INPDQ.R0000001
FOCSORT              1          SYS88229.TO95540.RA000.INPDQ.R0000002
OFFLINE              1          TSOINFOC.TSOINFOC.OFFLINE
SYSUT1               1          SYS88229.TO95058.RA000.INPDQ.SYSUT1
SYSEDIT              1          SYS88229.T095058.RA000.INPDO.EDIT
HOLD                 1          SYS88229.TO95058.RA000.INPDQ.R0000003
HOLDMAST             1          SYS88229.T095044.RA000.INPDQ.R0000004
CAR                  1          INPDQ.CAR.FOCUS
```

The DDNAME column lists all allocated file names (DDNAMEs). The OCCURRENCES column shows how many data sets are allocated to the corresponding DDNAME. This number will be one unless two or more data sets are concatenated under the DDNAME. The DSNAME column shows the fully qualified data set name (DSN) corresponding to the DDNAME. For concatenated data sets, all the data set names are shown.

Example: Displaying Attributes of a Queried DDNAME

This command displays attributes of the queried file name (DDNAME). If the specified DDNAME was not allocated, the command displays the attributes as either blanks or zeroes. Sample output is as follows:

```
>>? TSO DDNAME CAR
DDNAME              =          CAR
DSNAME              =          INPDQ.CAR.FOCUS
DISP                =          OLD
DEVICE              =          DISK
VOLSER              =          TSOPAK
DSORG               =          PS
RECFM               =          F
SECONDARY           =          1
ALLOCATION          =          TRACKS
BLKSIZE             =          4096
LRECL               =          4096
TRKTOT              =          2
EXTENTSUSED         =          1
BLKSPERTRK          =          10
TRKSPERCYL          =          15
CYLSPERDISK         =          886
BLKSWRITTEN         =          20
```

Syntax: How to Create a General List of Allocated Files

You can also create a general list of allocated files by DDNAME. To do so, specify a wildcard character (* or ?) with part of the DDNAME when you issue the query command. The syntax is

```
? {MVS|TSO} DDNAME ddn
```

where:

ddn

Is the first three characters of the DDNAME. You may specify up to eight characters, including the wildcard character. Use the wildcard character ? to represent one character. To represent a sequence of characters, use the * wildcard.

Example: TSO System Variables

The -? TSO DDNAME *ddname* statement is a Dialogue Manager control statement that works similarly to the ? TSO DDNAME *ddname* command, except that it places the information in variables instead of displaying the information on the terminal. These Dialogue Manager TSO system variables have the same names as the attributes returned by the ? TSO DDNAME *ddname* command. A complete list of variables is provided in *TSO DDNAME Variables* on page 198.

If there is no information for a variable, the variable will contain blanks if it is an alphanumeric variable, or zeroes if it is numeric.

The following is an example of how to use this Dialogue Manager control statement:

```
1. -? TSO DDNAME &DD.ENTER DDNAME.
2. -IF &DSNAME EQ ' ' GOTO ALLOCATE;
   -TYPE DATASET &DSNAME ALLOCATED TO &DD
   -EXIT
   -ALLOCATE
        .
        .
        .
```

The process is as follows:

1. This statement prompts you for the DDNAME. The information that is entered from the terminal is then placed in the variable &DD. Depending on the value of &DD, the system supplies the information for the variable &DSNAME. If no data set is allocated to the DDNAME that was supplied, the variable &DSNAME will contain a blank, as it is alphanumeric.

2. This statement tests whether the variable &DSNAME is a blank. If it is blank (that is, if no data set was found for the DDNAME entered by the operator), the procedure jumps to the label -ALLOCATE and continues. If it is not blank (that is, if a data set was found for the DDNAME entered by the operator), the procedure prints a message stating the name of the data set and exits.

Note: For detailed information on how to use Dialogue Manager control statements and variables, see the *Developing Applications* manual.

Reference: TSO DDNAME Variables

TSO DDNAME Variable	Meaning
&DDNAME	Queried file name (DDNAME).
&DSNAME	Fully qualified data set name. For concatenated data sets, only the first data set name is returned.
&DISP	Disposition: OLD, NEW, MOD, or SHR.
&DEVICE	DISK, TAPE, TERM, or READER-PRINTER.
&VOLSER	Disk or tape volume serial number(s).
&DSORG	Data set organization: PS (sequential); IS (indexed sequential); PO (partitioned); U (undefined); DA (direct).
&RECFM	Record format F, FB, V, VB, and so on.
&SECONDARY	Quantity specified for secondary allocations.
&ALLOCATION	Unit of secondary allocation: TRACKS, BLOCKS, or CYLINDER.
&BLKSIZE	Block size.
&LRECL	Record length, in bytes.
&TRKTOT	Total number of currently allocated tracks, spanning both primary and secondary allocation.
&EXTENTSUSED	Total number of currently allocated extents (max 16).

TSO DDNAME Variable	Meaning
&BLKSPERTRK	Number of blocks (of BLKSIZE bytes) that can be written on to 1 track of the device.
&TRKSPERCYL	Number of tracks per cylinder for the device.
&CYLSPERDISK	Number of cylinders per disk for the device.
&BLKSWRITTEN	Total number of blocks in the data set, assuming that all blocks are BLKSIZE long.

Syntax:

How to Determine If a Database Exists: ? TSO DSNAME

Since you can logically erase an existing data set by issuing a CREATE command against it if it already exists, it is necessary to have a means of testing for the existence of a data set before issuing the CREATE command. Use the following command:

```
? {TSO|MVS} DSNAME datasetname
```

You supply the data set name. If you supply only the unqualified data set name, FOCUS will take the prefix from the profile to create a fully qualified data set name. Do not specify member names.

This command may also be executed from Dialogue Manager, allowing you to test whether a data set exists:

```
-? TSO DSNAME dsname
```

In this case, there is no output message. The system variable &RETCODE is set and must be tested for the outcome. The possible results follow:

&RETCODE Value	Equivalent FOCUS Code/Message
0	(FOC488) Dataset is in catalog:
4	(FOC489) Dataset is in catalog, but not on volume indicated:
8	(FOC490) Dataset is not in catalog:

Procedure: How to Estimate Data Set Sizes to Determine Available Space

You can use the file attribute information returned by the ? TSO DDNAME command to determine the number of records in the data set and to estimate how much available space is left on the currently allocated tracks. Keep in mind that the estimates obtained from the formulas that follow are only approximations and do not take into account space that was reused after it was logically vacated by deleted segments.

- FOCUS data source formulas. The value for TOTAL PAGES is displayed as part of the output of the ? FILE *filename* command:

```
Available pages (without additional extents) = BLKSWRITTEN - TOTAL PAGES
BLKSWRITTEN = BLKSPERTRK x TRKTOT
```

- Fixed-block data set formulas:

```
RECSPERBLK = BLKSIZE/LRECL
No. of records = BLKSWRITTEN x RECSPERBLK
No. of free blocks = (TRKTOT x BLKSPERTRK) -BLKSWRITTEN
```

- Variable-length blocked data sets:

  The formulas for fixed-block data sets (that is, FOCUS SAVE files) usually apply to variable-length blocked (VB) data sets as well: VB data sets usually have the same length blocks, even though the description implies otherwise.

Note: If SET SHADOW=ON, use the following formula. The value for TOTAL PAGES is displayed as part of the output of the ? FILE *filename* command:

```
(TOTAL PAGES x 2) + 3
```

DYNAM Command

The FOCUS DYNAM command is used to manipulate data sets under z/OS. Although similar functions are available under TSO, the DYNAM command is very useful in non-TSO environments when TSO is not necessarily present. DYNAM is recommended instead of TSO commands to manipulate data sets.

Syntax: How to Manipulate Data Sets Under z/OS

```
DYNAM subcommand operand [operand]...
```

where:

subcommand

Required. Is one of the following operations:

`ALLOCATE`
`ALLOC`
`ALLO`
	Allocates a z/OS MVS data set or z/OS USS file. A wide variety of allocation options is supported.

`CONCAT` `CONC`	Concatenates data sets.
`FREE`	Frees data sets specified by ddnames or dsnames. Each name may contain wildcard characters.
`CLOSE` `CLO`	Closes data sets. Useful when the data sets cannot be freed because they are open. The command has the same syntax as DYNAM FREE above.
`COPY`	Copies an entire data set or selected PDS members. The command has powerful features such as record format conversion, either automatic or controlled by options.
`COPYDD`	Copies data between z/OS data sets and/or HiperFOCUS files. DYNAM COPY has been enhanced to handle all functions offered by COPYDD, and is recommended for use instead of COPYDD.
`DELETE` `DEL`	Deletes an entire data set or selected PDS members.
`RENAME` `REN`	Renames an entire data set or selected PDS members.
`SUBMIT` `SUB`	Submits z/OS jobs.
`COMPRESS` `COMP`	Compresses partitioned data sets (PDSs).
`SET TEMP`	Allocates temporary files.

`operand`

May be a keyword, a keyword followed by its parameter, or a parameter without a keyword.

The following rules apply to the DYNAM command:

- The subcommand name, keywords, and parameters are separated with one or more blanks. Keywords are coded in free format.

- A parameter may be a list of subparameters (for example, VOLUME for a multi-volume data set). Subparameters in the list should be separated with commas. For blanks between subparameters (with or without the comma), enclose the entire list in parentheses. For example:

  ```
  A,B (A,B) (A B) (A, B) (A,B C, D)
  ```

- DYNAM commands may span several lines. To do so, enter a hyphen (-) at the end of each line to be continued. In concatenating the lines, blanks after the hyphen and leading blanks from the next line are removed. Blanks before the hyphen are removed if they are preceded with a comma. The total length of a DYNAM command may not exceed 2048 characters.

- Most keywords may be truncated up to the shortest unambiguous length. The commonly used abbreviations are fixed. It is important to note that the unique truncation of a keyword may not always be valid as new keywords are added. Using the full keyword in files and using truncations interactively is recommended.

- Fixed abbreviations are listed in the following syntax descriptions. For example, DDNAME may be abbreviated as DD, DDN, DDNA, DDNAM, or DDNAME.

- Certain keywords have synonyms. For example, the keywords FILENAME and DDNAME are synonyms, and so are DATASET and DSNAME.

- As in TSO, a data set name can be enclosed in single quotation marks. Prefix substitution is not supported; only the fully qualified data set names should be specified.

- Some DYNAM commands accept either the ddname or data set name (dsname) as the same parameter. In such cases, the parameter is considered a ddname if it is not longer than 8 bytes, does not contain periods (.), and is not enclosed in single quotation marks. Otherwise, the parameter is considered a data set name. Thus, to specify an unqualified data set name, enclose it in single quotation marks.

- Dialogue Manager variables can be used in a DYNAM command, in which case the variable value is substituted at run time. If a variable is used as part of a data set name, it may need to be followed by extra period characters (.) in order to be substituted correctly. For example, in the following DYNAM command, three periods are required. The first period ends the variable name. If there were only a second period, it would be considered an indexed variable, and no period would be inserted in the string. The third period indicates that one period should be inserted into the string after the variable name:

  ```
  -SET &OUT_NAME = CAR ;
  DYNAM ALLOC F CAR DA USER1.&OUT_NAME...FOCUS SHR REU
  ```

The resulting string is:

```
DYNAM ALLOC F CAR DA USER1.CAR.FOCUS SHR REU
```

The DYNAM command and its subcommands, except for COMPRESS and CLOSE, are available from the DYNAM Utilities Menu. Additional allocations for the menu are not required. Private applications may be included on the primary menu. To access the menu, enter the command:

```
EX DYMENU
```

Use of Data Sets

z/OS obtains a lock for any allocated data set name: a shared lock for those specified as SHR; an exclusive lock for OLD, NEW, or MOD.

Although data sets can be allocated more than once in a job step, only one type of lock may be obtained. For example, if the data set is allocated as SHR and is then allocated as OLD in the same step, the z/OS lock changes from shared to exclusive, and the data set will not be available for use by other jobs until all allocations in this job are freed.

In addition, a z/OS exclusive lock is not sufficient in multi-user environments, such as FOCUS MSO. If one user were to allocate a data set as OLD, z/OS would allow another user to allocate it as OLD also, because both requests occur in the same job step. This can cause data set corruption due to simultaneous updating.

In order to alleviate these issues, the DYNAM commands that manipulate data sets use an improved locking mechanism, similar to that implemented in ISPF:

- Any output PDS is allocated by DYNAM (or preallocated by the user) as SHR. This avoids exclusive z/OS locking, which lasts until all data set allocations are free.
- To protect from simultaneous updating, DYNAM obtains an exclusive lock as used by ISPF (and other programs, including LINKEDIT), but only during the actual update time. This lock controls access to the data set between users, even from within the same job step.

  Note: The DYNAM locking mechanism protects from simultaneous updating and possible corruption of data, but does not protect from updating and simultaneous reading. For example, it is possible to continue to read a PDS member recently deleted by another user.

ALLOCATE Subcommand

The DYNAM ALLOCATE command allocates a data set.

Syntax: How to Allocate a z/OS MVS Data Set With the ALLOCATE Subcommand

```
DYNAM ALLOCATE normal_disp [CLOSE]

DDNAME ddname [DEFER] [LONGNAME lnam] [DSNAME dsname{(memname)|(n)}
[DUMMY]
[EXPDT date]
[HIPER OFF]
[DALDKYL labelname]

[INPT|OUTPT]

[LABEL type]
[MEMBER memname] status [MSVGP msvgp]

[PARALLEL] [PASSWORD password] [PERM] [POSITION nnnn]
[REFVOL dsname] [RETPD days] [REUSE]
[UNIT unit [UCOUNT n]]
[VOLUME volser]
```

Space operands are:

```
space_format

space_alloc

 [DIR n]
 [PRIMARY n1]
 [RELEASE] [ROUND]
 [SECONDARY n2] [SPACE space]
```

DCB operands are:

```
 [BLKSIZE n] [BUFNO n]
 [DEN n] [DSORG dsorg]
 [LRECL n]
 [EATTR {NO|OPT}]
 [RECFM recfm] [REFDD ddname] [REFDSN dsname]
```

SMS and VSAM operands are:

```
 [DATACLASS name] [DSNTYPE {LIBRARY|PDS|EXTREQ|EXTPREF}]

 [KEYOFF n]
 [LIKE dsname]
 [MGMTCLASS name]
 [RECORG recorg]
 [SECMODEL name] [STORCLASS name]
 [BUFND m]
 [BUFNI n]
```

Output printing operands are:

```
[DEST dest[.user]]

[FCB name [ALIGN|VERIFY]] [FORMS name]

[HOLD]
[OUTLIM n] [OUTPUT name]
[SYSOUT class]
[USER user]
[WRITER name]
```

where:

`ALLOCATE`

Can be abbreviated as ALLOC or ALLO.

`normal_disp`

Can be one of the following:

`CATALOG`
`DELETE`
`KEEP`
`UNCATALOG`
`UNCAT`
	Data set normal disposition. By default, for a data set status of NEW, if *dsname* is specified, the disposition is CATALOG. Otherwise, the disposition is DELETE. Incompatible with SYSOUT. Synonyms are: CATALOG-CATLG.

`CLOSE`

Deallocation of the data set at close rather than at the end of the step. JCL analogy: FREE=CLOSE.

`DDNAME` *`ddname`* `DDDEFER`

DDNAME to be associated with an allocation. Must be specified. Synonyms are: DDname-Filename. Assign device(s) to the data set but defer mounting of the volume(s) until the data set is opened. JCL analogy: DEFER in UNIT.

`DSNAME` *`dsname`* `[(`*`memname`*`)]`

Member name can be specified either in parentheses after dsname or using keyword MEMBER (see below). If dsname is specified as asterisk (*), terminal is allocated. This is used for output only. Synonyms are: DSname-DAtaset.

`[(`*`n`*`)]`

Relative GDG number.

`DUMMY`

Dummy data set is to be allocated.

`DALDKYL` *`labelname`*

Specifies the label for the encryption key used by the system to encrypt the data set. The maximum length of the label is 64 characters. The TSO ALLOC equivalent is DSKEYLBL.

For example, to allocate a DD with pervasive encryption, key MYLABELNAME:

```
DYNAM ALLOC DD ASDF .... DALDKYL MYLABELNAME ... SHR
```

This keyword has an effect only when creating an extended format data set. the first value for the DISP keyword must be coded or defaulted.

`EXPDT` *`date`*

Expiration date in format YYDDD, YYYY/DDD, or YYYYDDD. Incompatible with RETPD and SYSOUT.

`HIPER OFF`

Prohibit allocation in a hiperspace. Equivalent to the "UNIT NOHIPER" and is used when UNIT is to be specified too. For example, "UNIT VIO HIPER OFF".

`INPT OUTPT`

Data set is to be processed as input only (INPT) or output only (OUTPUT). JCL analogy: IN in LABEL. Incompatible with SYSOUT.

`LABEL` *`type`*

Type of volume labels. Can be one of the following: NL, SL, NSL, SUL, BLP, LTM, AL, or AUL. Incompatible with SYSOUT.

`LONGNAME` *`lnam`*

Is a Master File name longer than eight characters. Creates a copy of the corresponding short Master File in the PDS allocated to DD HOLDMAST. For more information, see the *Describing Data* manual.

`MEMBER` *`memname`*

Name of a PDS member to be allocated. See also DSNAME.

`status`

Data set status. Default: NEW. Incompatible with SYSOUT. Can be one of the following: `MOD, NEW, OLD, SHR`

`MSVGP` *`msvgp`*

Identification of a group of mass storage system (MSS) virtual volumes. Incompatible with SYSOUT and VOLUME.

PARALLEL

Each volume is to be mounted on a separate device. JCL analogy: P in UNIT.

PASSWORD *password*

Password for a password protected data set.

PERM

The allocation is to be permanent—that is, protected from being freed or concatenated by any DYNAM command issued by an MSO user. The operand is valid only in an MSO server initialization profile.

POSITION *nnnn*

Data set sequence number on a tape volume, up to 9999. JCL analogy: the first subparameter in LABEL.

REFVOL *dsname*

Volume serial information is to be obtained from the specified cataloged data set. JCL analogy: VOL=REF=dsname. Incompatible with SYSOUT and VOLUME.

RETPD *days*

Retention period, up to 9999 days. Incompatible with EXPDT and SYSOUT.

REU[SE]

If the ddname to be allocated is already in use, it is to be freed.

UNIT *unit*

Device group name, device type, specific unit address, or NOHIPER. NOHIPER prohibits allocation in a hiperspace, meaningful for a temporary (NEW,DELETE) data set; see also HIPER OFF.

UCOUNT *n*

Number of volumes to allocate.

VOLUME *volser* VOL

Volume serial numbers. Incompatible with REFVOL and SYSOUT. Synonyms are: VOLume-VOLser.

Space operands may be:

space_format

Can be one of the following:

ALX CONTIG MXIG	Format of the primary space to be allocated: up to five contiguous areas (ALX); one contiguous area (CONTIG); one maximal contiguous area (MXIG). JCL analogy: ALX/CONTIG/MXIG in SPACE.

space_alloc

Can be one of the following:

BLOCKS [n] CYLINDERS MEGABYTES PAGES TRACKS	N represents units of primary and secondary space allocation. If parameter for BLOCKS is omitted, the average block length is copied from BLKSIZE. If space unit is omitted but SPACE and BLKSIZE are specified, BLOCKS equal BLKSIZE is used. For PAGES, BLOCKS 4096 is used. BLKSIZE must be specified if the BLOCKS parameter is specified. Synonyms are: CYLINDERS-CYLs, TRACKS-TRKs.

DIR *n*

Number of 256-byte records for the directory of a PDS.

PRIMARY *n1*

Primary space quantity. See also SPACE.

RELEASE

Unused space is to be released when the data set is closed. Synonyms are: RELEASE-RLSE.

ROUND

If space is requested in BLOCKS, MEGABYTES, or PAGES, it is to be rounded to whole cylinder(s).

SECONDARY *n2*

Secondary space quantity. See also SPACE.

`SPACE` *space*

`SP`

Primary (*n1*) and/or secondary (*n2*) space quantity in one of the following formats:

n1`/(`*n1*`)/`*n1*`,`*n2*`/(`*n1*`,`*n2*`)/`*n1* *n2*`/(`*n1* *n2*`)/,`*n2*`/(,`*n2*`)`

See also PRIMARY and SECONDARY.

DCB operands may be:

`BLKSIZE` *n*

Block size, up to 32760. See also BLOCKS.

`BUFNO` *n*

Number of buffers, up to 255.

`DEN` *n*

n represents magnetic tape density: 0, 1, 2, 3, or 4 for 200, 556, 800, 1600, 6250 bpi respectively.

`DSORG` *dsorg*

Data set organization. Default, for NEW only; PO if DIR or DSNTYPE specified; PS otherwise. The following values are syntactically correct:

`VS` VSAM.

`PO/POU` PDS or PDS unmovable.

`DA/DAU` direct access or direct access unmovable.

`PS/PSU` physical sequential or physical sequential unmovable.

`LRECL` *n*

Logical record length, up to 32760.

`EATTR {NO|OPT}`

Specifies if the data set to be created on a DASD volume can have extended attributes and reside in the extended address space (cylinder-managed space on the volume beyond the 64k cylinder limit). Valid values are:

- **NO.** Data set to be created is not allowed to have extended attributes. it must not be created in the extended address space. This is the default value.
- **OPT.** Data set to be created allows extended attributes to be stored in a 'FORMAT 8/9 DSCB' on the volume. The data set is allowed to be created in the extended address space. For example:

```
DYNAM ALLOC DD EAV DSN hlq.EAV.DATA NEW DSORG PS RECFM FB LRECL 80
CYL EATTR OPT
```

RECFM *recfm*

Record format. The first letter should be D, F, U, or V, which may be followed by any valid combination of A, B, M, S, or T:

A records with ISO/ANSI control characters.

B blocked records.

D variable-length ISO/ANSI tape records.

F fixed-length records.

M records with machine code control characters.

S standard fixed-length or spanned variable-length records.

T track overflow.

U undefined-length records.

V variable-length records.

REFDD *ddname*

DCB attributes are to be copied from the specified ddname. Under TSO, EXPDT and INPT/OUTPT specifications are also copied. Any of those can be overridden by appropriate keyword on the same command. JCL analogy: DCB=*.ddname. Incompatible with REFDSN.

REFDSN *dsname*

DCB attributes (DSORG, RECFM, OPTCD, BLKSIZE, LRECL, RKP, KEYLEN) and EXPDT are to be copied from the specified cataloged data set. Any of those can be overridden by appropriate keyword on the same command. JCL analogy: DCB=dsname. Incompatible with REFDD.

SMS and VSAM operands are:

DATACLASS *name*

Name of a data class for an SMS managed data set.

DSNTYPE {LIBRARY|PDS|EXTREQ|EXTPREF}

LIBRARY is for a new partitioned extended (PDSE), and PDS is for a new partitioned data set. A PDSE cannot contain load modules, should be SMS-managed, and allows concurrent updating of different members.

EXTREQ specifies that a data set must be extended format (required). It may be VSAM or sequential.

EXTPREF requests that the data set be extended format (preferred). If the system cannot create it in extended format, it will try basic format. It may be VSAM or sequential.

`KEYOFF` *n*

Offset of the key in each logical record for a new VSAM key-sequenced (RECORG KS) data set.

`LIKE` *dsname*

Allocation attributes (DSORG, RECORG or RECFM, LRECL, KEYLEN, KEYOFF, SPace, DIR) are to be copied from the specified cataloged data set (model). Any of those can be overridden by appropriate keyword on the same command.

`MGMTCLASS` *name*

Name of a management class for an SMS managed data set.

`RECORG` *recorg*

VSAM record organization: KS, ES, RR, or LS for key-sequenced, entry-sequenced, relative record, or linear space data set respectively.

`SECMODEL` *name*

Data set RACF profile is to be copied from the named existing RACF profile.

`STORCLASS` *name*

Name of a storage class for an SMS managed data set.

`BUFND` *m*

Is the number of VSAM DATA buffers.

`BUFNI` *n*

Is the number of VSAM INDEX buffers.

Output printing operands may be:

`DEST` *dest*[*.user*]

Remote destination for a SYSOUT data set. In conjunction with user ID, it is a node and a user at that node; the user ID can be coded after the period (.) or using the USER keyword (see below).

`FCB name [ALIGN|VERIFY]`

Name of an FCB (forms control buffer) image to be used for printing of a data set. The operator may be asked to check the printer forms alignment (ALIGN), or to verify the FCB image name displayed on the printer (VERIFY).

`FORMS name FORM`

A SYSOUT form name. JCL analogy: third subparameter in SYSOUT, FORMS in OUTPUT JCL.

`HOLD`

A SYSOUT data set is to be placed on the hold queue.

`OUTLIM n`

Limit for the number of logical records in a SYSOUT data set.

`OUTPUT name`

Name(s) of OUTPUT JCL statement(s) to be associated with a SYSOUT data set.

`SYSOUT class`

A SYSOUT data set is to be allocated and the specified output class (A-Z, 0-9) is to be assigned. If asterisk (*) or NULL is coded, the class is copied either from CLASS in OUTPUT JCL if it is specified, or from MSGCLASS in JOB otherwise.

`USER user`

A SYSOUT data set is to be routed to the specified user ID. DEST (see above) is required to specify a user's node.

`WRITER name`

Name of an installation written system output printing routine. JCL analogy: second subparameter in SYSOUT. Incompatible with USER.

In addition to the shown fixed abbreviations and synonyms, keywords may be abbreviated up to the unique truncation. Those abbreviations are not fixed and may be changed when new keywords are added. They may be used interactively to save some keystrokes, but when a command is saved in a file, using unabbreviated keywords is recommended.

Example: Using the DYNAM ALLOCATE Command

Allocate an existing data set:

```
DYNAM ALLOC DD MYDD DS MYID.DATA.SET SHR REU
```

Allocate a new data set. Defaults: NEW, CATALOG (DSname present), DSORG PO (not-zero DIR present):

```
DYNAM ALLOC DD MYDD DS MYID.DATA.SET SPACE 6,2 TRACKS DIR 4 UNIT SYSDA -
RECFM FB LRECL 80 BLKSIZE 1600
```

Allocate a terminal:

```
DYNAM ALLOC DD MYDD DS *
```

Allocate a SYSOUT data set with default output class. Upon freeing, the data set is sent to the user ID U1234 at node SYSVM:

```
DYNAM ALLOC DD MYDD SYSOUT * DEST SYSVM.U1234
```

Syntax: **How to Allocate a z/OS USS File With the ALLOCATE Subcommand**

```
DYNAM ALLOCATE DDNAME ddname PATHNAME path REUSE -
    RECFM VB LRECL lrecl [other_operands]
```

where:

ddname
Is the logical name to be associated with the allocation.

path
Identifies a UNIX file.

A pathname begins with a slash (/). The system treats any consecutive slashes as a single slash. The pathname can be 2 to 250 characters long, including the initial slash.

A pathname consists of the names of the directories from the root to the file being identified, and the name of the file, including the extension. The pathname is case sensitive. The form is:

/name1/name2/.../fname.ext

For example:

```
/u/user1/hcar1.mas
```

The pathname consists of printable characters from X'40' to X'FE'. A filename can contain characters outside this range,but these characters cannot be specified in the JCL. Enclose the pathname in single quotation marks if it contains any character other than the following characters:

- Uppercase letters.

- Numeric digits.
- Special characters (#, $, or @).
- Slash (/).
- Asterisk (*).
- Plus sign (+).
- Hyphen (-).
- Period (.).
- Ampersand (&).

Note: Allocation verifies the validity of the pathname. However, there is no ENQ or locking of the pathname, so it is possible to modify a pathname component, even in an asynchronous process. Doing this may cause errors in OPEN or unexpected results, with no errors reported.

`REUSE`
Specifies that the file name being allocated is to be freed and reallocated if it is currently in use.

`RECFM VB`
Identifies the record format. For a USS file, the record format is VB.

`LRECL` *`lrecl`*
Defines the record length. The record length must be at least as large as the longest input line.

The following operands are supported with PATHNAME:

- BLKSIZE
- BUFNO
- DSNTYPE
- DUMMY
- FILEDATA
- LRECL
- NCP
- PATHDISP

- PATHMODE
- PATHOPTS
- RECFM
- REUSE
- TERM

The syntax for the optional supported operands follows.

`BLKSIZE n`

Block size, up to 32760.

`BUFNO n`

Number of buffers, up to 255.

`DSNTYPE {HFS|PIPE}`

HFS specifies a UNIX file system.

PIPE specifies a first-in first-out (FIFO) special file, which is also called a named pipe.

`DUMMY`

Dummy data set is to be allocated.

`FILEDATA(content_type)`

Controls the data conversion method, performed by the network file system client, when accessing network files on a different system. The other system might be OS/390®, AIX®, or certain other kinds of systems. The FILEDATA keyword is used to describe the content type of a z/OS UNIX file so that the system can determine how to process the file. Valid content types are:

- **BINARY.** Specifies that the file described by the DYNAM command is a byte-stream file and does not contain record delimiters. The access method does not insert or delete record delimiters.

  If you do not code the FILEDATA operand, the system assigns a default value of BINARY to the UNIX file.

- **TEXT.** Specifies that the file described by the DYNAM command contains records delimited by the EBCDIC newline character (x'15').

  See the appropriate DFSMS/MVS publications for more details about the Network File System client and its conversion methods.

  You need to code the PATH operand together with the FILEDATA operand.

You can code the FILEDATA operand together with the following ALLOCATE operands: BLKSIZE, BUFNO, DSNTYPE, DUMMY, LRECL, NCP, PATHDISP, PATHMODE, PATHOPTS, RECFM.

- **RECORD.** Indicates that the data consists of records with prefixes. The record prefix contains the length of the record that follows. On output, the access method inserts a record prefix at the beginning of each record. On input, the access method uses the record prefix to determine the length of each record. The access method does not return the prefix as part of the record. Code FILEDATA(RECORD) when you cannot code FILEDATA(TEXT) because your data might contain bytes that are considered delimiters.

  Note: The record prefix for FILEDATA(RECORD) is mapped by the IGGRPFX macro. This is different from the record descriptor word (RDW) that is in z/OS physical sequential format-V data sets.

NCP(*number_of_channel_programs*)

Specifies the maximum number of READ or WRITE macro instructions allowed before a CHECK or WAIT macro instruction is issued. The maximum number must not exceed 255 and must be less than 255 if the address space does not have enough virtual storage. If you are using chained scheduling, you must specify an NCP value greater than 1. If you omit the NCP operand, the default value is 1.

PATHDISP([*normal_disposition*] [, *abnormal_disposition*])

Specifies the disposition of a UNIX file upon normal and abnormal (conditional) TSO/E session termination. Valid values are the same for both:

- **KEEP.** Specifies that the file should be kept. This is the default values.
- **DELETE.** Specifies that the file should be deleted.

PATHMODE(*file_access_attribute*)

Specifies the file access attributes when the PATHOPTS operand also specifies OCREAT.

If you specify either OCREAT alone, or OCREAT and OEXCL, in the PATHOPTS operand, and if the file does not exist, MVS performs an open() function. The options from PATHOPTS, the pathname from the PATHNAME operand, and the options from PATHMODE (if specified) are used in the open(). MVS uses the close() function to close the file before the application program receives control.

For status group options other than OCREAT and OEXCL, it is assumed that the application or OPEN macro passes the operands to the open() function without modification. That is, this application or OPEN macro uses dynamic allocation information retrieval (the DYNALLOC macro) to retrieve the subparameters specified for PATHOPTS, and passes the subparameters to the open() function. The application program can ignore or modify the information specified in the JCL or on the ALLOCATE command.

You can specify up to 14 file access attributes. Separate each with a comma. The system treats duplicate specifications as a single specification. Subparameter Definition:

SIRUSR
: Specifies permission for the file owner to read the file.

SIWUSR
: Specifies permission for the file owner to write the file.

SIXUSR
: Specifies permission for the file owner to search, if the file is a directory, or to execute, for any other file.

SIRWXU
: Specifies permission for the file owner to read, write, and search, if the file is a directory, or to read, write, and execute, for any other file.

  This value is the bit inclusive OR of SIRUSR, SIWUSR, and SIXUSR.

SIRGRP
: Specifies permission for users in the file group class to read the file.

SIWGRP
: Specifies permission for users in the file group class to write the file.

SIXGRP
: Specifies permission for users in the file group class to search, if the file is a directory, or to execute, for any other file.

SIRWXG
: Specifies permission for users in the file group class to read, write, and search, if the file is a directory, or to read, write, and execute, for any other file.

  This value is the bit inclusive OR of SIRGRP, SIWGRP, and SIXGRP.

SIROTH
: Specifies permission for the users in the file other class to read the file.

SIWOTH
: Specifies permission for the users in the file other class to write the file.

SIXOTH
: Specifies permission for users in the file other class to search, if the file is a directory, or to execute, for any other file.

SIRWXO
: Specifies permission for users in the file other class to read, write, and search, if the file is a directory, or to read, write, and execute, for any other file.

  This value is the bit inclusive OR of SIROTH, SIWOTH, and SIXOTH.

`SISUID`
Specifies that the system set the user ID of the process to be the same as the user ID of the file owner when the file is run as a program.

`SISGID`
Specifies that the system set the group ID of the process to be the same as the group ID of the file owner when the file is run as a program. The group ID is taken from the directory in which the file resides.

When creating a new UNIX file, if you do not code a PATHMODE operand on a DYNAM command with a PATHNAME operand, the system sets the permissions to zero, which prevents access by all users. If the UNIX file already exists, PATHMODE is checked for syntax, but ignored. The permission bits are left as they are set.

`PATHOPTS(`*`file_options`*`)`
Specifies the file access and status used when accessing a file specified in the PATHNAME operand. You can specify up to seven file options. Separate each with a comma. The system treats duplicate specifications as a single specification.

`Access Group Options`
Specify only one of the following options.

Note: If you specify more than one Access Group, the system ignores them and uses ORDWR.

- **ORDONLY.** Specifies that the program can open the file for reading.
- **OWRONLY.** Specifies that the program can open the file for writing.
- **ORDWR.** Specifies that the program can open the file for reading and writing. Do not use this option for a FIFO special file.

`Status Group Options`
You can specify up to 6 of these options.

For status group options other than OCREAT and OEXCL, the description in this book assumes that the application or OPEN macro passes the operands to the open() function without modification. That is, this application or OPEN macro uses dynamic allocation information retrieval (the DYNALLOC macro) to retrieve the subparameters specified for PATHOPTS and passes the subparameters to the open() function. The application program can ignore or modify the information specified in the JCL or on the ALLOCATE command.

- **ORDONLY.** Specifies that the program can open the file for reading.
- **OWRONLY.** Specifies that the program can open the file for writing.

- **ORDWR.** Specifies that the program can open the file for reading and writing. Do not use this option for a FIFO special file.
- **OAPPEND.** Specifies that the system sets the file offset to the end of the file before each write, so that data is written at the end of the existing file.
- **OCREAT.** Specifies that the system is to create the file. If the file already exists, the operation fails if OEXCL is specified, and opens the existing file if OEXCL is not specified.
- **OEXCL.** Specifies that, if the file does not exist, the system is to create it. If the file already exists, open() fails. Note that the system ignores OEXCL if OCREAT is not also specified.
- **ONOCTTY.** Specifies that, if the PATHNAME operand identifies a terminal device, open() does not also make the terminal device the controlling terminal of the process and the session.
- **ONONBLOCK.** Specifies the following, depending on the type of file:
  - For a FIFO special file with ORDONLY option set, ONONBLOCK specifies read-only opening of the file. If ONONBLOCK is not specified, the read-only open() blocks until a process opens the file for writing.
  - For a FIFO special file with OWRONLY option set, ONONBLOCK specifies that the system immediately process a request for a write-only open() of the file, if a process has already opened the file for reading. If the file is not open for reading, the system returns an error. If ONONBLOCK is not specified, the write-only open() blocks until a process opens the file for reading.
  - For a character special file that supports a nonblocking open(), ONONBLOCK specifies that the system immediately returns if it cannot open a file because the device is not ready or available. If ONONBLOCK is not specified, the open() blocks until the device is ready or available.

    Specifications of ONONBLOCK has no effect on other file types.
- **OSYNC.** Specifies that the system is to move data from buffer storage to permanent storage before returning control from a callable service that performs a write.

- **OTRUNC.** Specifies that the system is to truncate the file to zero length if all of the following conditions are true:
  - The file specified on the PATH operand exists.
  - The file is a regular file.
  - The file successfully opened with ORDWR or OWRONLY.

  The system does not change the mode and owner. OTRUNC has no effect on FIFO special files or directories.

`TERM`
Indicates to the system that a data set is coming from or going to a terminal for a TSO/E user.

CONCAT Subcommand

The DYNAM CONCAT command concatenates up to 16 data sets.

Syntax: How to Concatenate Data Sets

```
DYNAM CONCAT [PERM] DDNAME ddname1 ddname2 [ddname3...]
```

where:

`CONCAT`

Can be abbreviated as CONC.

`PERM`

Optional. Marks the concatenation as permanent—that is, protected from being freed or concatenated again by any DYNAM command issued by an MSO user. Valid only in an MSO server initialization profile.

`DNAME`

`DDN`

`DD`

Required. Synonym is FILENAME.

`ddname1`

Is the first ddname to be concatenated and associated with the resulting concatenated group.

ddname2

Is the second and any subsequent ddname to be concatenated.

Example: Using the DYNAM CONCAT Command

```
DYNAM CONCAT DDN FOCEXEC MYEX NEWEX
```

FREE Subcommand

The DYNAM FREE command deallocates any number of specified data sets.

Syntax: How to Deallocate Data Sets

```
DYNAM FREE {DDNAME ddname [ddname...]|DSNAME dsname [dsname...]}
DYNAM FREE LONGNAME lnam
```

where:

DDNAME

DDN

DD

Is required if there is no dsname. Synonym is FILENAME.

ddname

Is the ddname of the data set to be freed.

DSNAME

DSN

DS

Is required if there is no ddname. Synonym is DATASET.

dsname

Is the name of the data set to be freed. All ddnames associated with this dsname, except concatenated groups, are deallocated.

lnam

Is a Master File name longer than eight characters. For more information, see the *Describing Data* manual.

While at least one ddname or data set name is required, you may specify more than one ddname or data set name. Each specified name may contain asterisks (*) and question marks (?) as wildcards. Wildcards are special characters used to specify a subset of names rather than one name. The wildcards can appear anywhere in a name and mean the following:

*

Represents any number of characters. For example, "*Q*" matches any name containing the character "Q".

?

Represents any single character. For example, "?Q?" matches any 3-character name containing the character "Q" in the middle.

If the ddname is not found, an error message is issued only if a single ddname without wildcards is specified. An error message is not displayed if a data set or more than one ddname is not found.

Example: Using the DYNAM FREE Command

```
DYNAM FREE DDN SYS0* TEMP?
DYNAM FREE DSN MYID.DATA.SET
```

CLOSE Subcommand

The DYNAM CLOSE command closes data sets, which cannot be freed because they are open.

Syntax: How to Close Data Sets

```
DYNAM CLOSE {DDNAME ddname [ddname...]|DSNAME dsname [dsname...]}
```

where:

CLOSE

Can be abbreviated as CLO.

DDNAME

DDN

DD

Is required if there is no dsname. Synonym is FILENAME.

ddname

Is the ddname of the data set to be closed.

DSNAME

DSN

DS

Is required if there is no ddname. Synonym is DATASET.

dsname

Is the name of the data set to be closed. All ddnames associated with this dsname, except concatenated groups, are closed.

While at least one ddname or data set name is required, more than one ddname or data set name may be specified. Each specified name may contain wildcard characters. The same rules apply to the DYNAM CLOSE command as for the DYNAM FREE command (see *FREE Subcommand* on page 221).

COPY Subcommand

The DYNAM COPY command copies an entire z/OS data set or selected PDS members.

Syntax: How to Copy Entire z/OS Data Sets or Selected PDS Members

```
DYNAM COPY dname1 {[TO] dname2 [[MEMBER] members]|[MEMBER] members}
 [options]
```

where:

dname1

Is the dsname or ddname of the input data set. This is a positional parameter. It must precede all other operands.

TO

May be omitted if *dname2* does not match a reserved word, the MEMBER keyword, an option, or the TO keyword. To avoid confusion, use the TO keyword whenever *dname2* is a ddname.

dname2

Is the dsname or ddname of the output data set. If the output data set is not a PDS and the dsname is specified, it will be allocated as OLD. If the ddname is specified, and the status is SHR, you must make sure that other users do not access the data set during COPY. Unlike ISPF, DYNAM will lock a non-PDS data set in order to prevent simultaneous updating by different DYNAM users.

MEMBER

May be omitted if members are specified in parentheses.

members

Can be a single member specification or a list of member specifications. If the members are enclosed in parenthesis, blanks preceding the left parenthesis may be omitted.

options

May be one or more of the following options:

APPEND	Adds the input to the end of the existing data, if the output is a sequential data set.
FORCE	Copies input DCB attributes (RECFM, BLKSIZE, LRECL and KEYLEN) to the output data set. By default, only missing values are assigned. **Note:** Do not use FORCE to change the LRECL of one member of an existing PDS as doing so may potentially destroy the PDS.
KEYMOD	Allows key modification according to input/output KEYLEN: truncation or padding with binary zeros.
REPLACE	Replaces all output members matching the selected member names.
TRUNCATE	Allows truncation of input records that are longer than the output record length. Since trailing blanks are truncated automatically when RECFM is different, the keyword is used either to cut records of the same format or to cut non-blank data.

A member specification has the following syntax

mem[,[*newmem*][,REPLACE]]

where:

mem

Is the selected member name.

newmem

Is the optional new name for the output member.

REPLACE

Is optional and specifies an existing member to be replaced in the output PDS.

Since the comma may be used in member specifications, they are separated with one or more blanks when specified in a list. Therefore, a list of member specifications is always enclosed in parentheses. For example:

```
(MEM MEM,NEWMEM MEM,NEWMEM,R MEM,,R)
```

Note:

- All conversions between different DCB attributes (RECFM, BLKSIZE, and LRECL) are performed automatically.
- If the entire PDS is copied or any selected member's directory entry contains a TTRN in user data (for example, a load module), the IBM utility IEBCOPY is invoked. In this case, all options except REPLACE are ignored, format conversion is not possible, and copying members to the same PDS is not supported. Note that IEBCOPY requires APF authorization in order to be performed.
- If the main member and its alias names are copied, their relationship remains the same on the output PDS. Aliases are not supported for members of HiperFOCUS files.
- If a specified ddname has been allocated with a member name, the data set is treated as sequential. However, if input and/or output is a HiperFOCUS file, member name(s) overriding the allocated one can be specified in the command.
- If the entire PDS is to be copied (no member list was provided), all members are replaced, regardless of the presence or absence of the REPLACE keyword.

Example: Using the DYNAM COPY Command

Copies the entire data set, whether it is a PDS or not.

```
DYNAM COPY MYDD MYID.DATA.SET
```

All four commands are equivalent. Either input or output may be a sequential data set, or both are PDSs.

```
DYNAM COPY MYDD MYID.DATA.SET MEMBER MEM
DYNAM COPY MYDD MYID.DATA.SET(MEM)
DYNAM COPY MYDD(MEM) MYID.DATA.SET
DYNAM COPY MYDD MEMBER MEM MYID.DATA.SET
```

Copies and renames one member.

```
DYNAM COPY MYID.DATA.LIB TO MYDD(MEM1,MEM2)
```

Copies two members.

```
DYNAM COPY MYID.DATA.LIB TO MYDD(MEM1 MEM2)
```

Copies two members into same PDS with renaming.

```
DYNAM COPY MYDD(OLD1,NEW1,R OLD2,NEW2)
DYNAM COPY MYDD(OLD1,NEW1 OLD2,NEW2) REPL
```

COPYDD Subcommand

The DYNAM COPYDD command copies a sequential data set, a PDS member, or a HiperFOCUS file.

Syntax: How to Copy Sequential Data Sets, PDS Members, or HiperFOCUS Files

```
DYNAM COPYDD ddname1[(mem1)] ddname2[(mem2)]
```

where:

ddname1

Is the ddname of the input data set.

mem1

Optional. Is the input member name.

ddname2

Is the ddname of the output data set.

mem2

Optional. Is the output member name.

Note:

- If the specified ddname has been allocated with a member name, the data set is treated as sequential. However, if input and/or output is a HiperFOCUS file, a member name overriding the allocated one can be specified in the command.
- Identically named members are always replaced on the output PDS.
- All conversions between different DCB attributes (RECFM, BLKSIZE, and LRECL) are performed automatically.
- Since the DYNAM COPY command has been upgraded to work with HiperFOCUS files and has more features than COPYDD, using COPY instead of COPYDD is recommended.

Example: **Using the DYNAM COPYDD Command**

MYDD1 is a sequential file or is allocated with a member name. If MYDD2 is allocated with a member name, MEM2 is valid only if at least one of the ddnames is a HiperFOCUS file.

```
DYNAM COPYDD MYDD1 MYDD2(MEM2)
```

DELETE Subcommand

The DYNAM DELETE command deletes an entire z/OS data set or selected PDS members.

Syntax: **How to Delete an Entire z/OS Data Set or Selected PDS Members**

The syntax to delete an entire z/OS data set is:

```
DYNAM DELETE dsname
```

To delete individual members use:

```
DYNAM DELETE dname [MEMBER] members
```

where:

`DELETE`

Can be abbreviated as DEL.

`dsname`

Is the data set name to be deleted and uncataloged.

`dname`

Is the dsname or ddname of a PDS containing one or more members to be deleted. The ISPF-like lock is obtained.

`MEMBER`

May be omitted if the members are specified in parentheses.

`members`

Can be a single member name or a list of members. If the members are enclosed in parentheses, blanks before the left parenthesis can be omitted.

Example: **Using the DYNAM DELETE Command**

```
DYNAM DELETE MYID.DATA.OLD
DYNAM DEL MYID.DATA.LIB MEMBER OLD1,OLD2
DYNAM DELETE MYDD(OLD1,OLD2)
DYNAM DEL MYDD(OLD1 OLD2 OLD3)
```

RENAME Subcommand

The DYNAM RENAME command renames an entire z/OS data set or selected PDS members.

Syntax: How to Rename an Entire z/OS Data Set or Selected PDS Members

The syntax to rename an entire z/OS data set is:

```
DYNAM RENAME dsname1 dsname2
```

To rename individual members, use

```
DYNAM RENAME dname [MEMBER] members [REPLACE]
```

where:

`RENAME`

Can be abbreviated as REN.

`dsname1`

Is the data set name to be renamed and uncataloged.

`dsname2`

Is the new name to be assigned to the data set and cataloged.

`dname`

Is the dsname or ddname of a PDS containing one or more members to be renamed. The ISPF-like lock is obtained.

`MEMBER`

May be omitted if the members are specified in parentheses.

`members`

Can be a single member specification or a list of members. If the members are enclosed in parentheses, blanks before the left parenthesis can be omitted.

`REPLACE`

Optional. Replaces all members matching the specified new names.

A member specification has the following syntax:

`oldmem,newmem``[,REPLACE]`

where:

oldmem

Is the original member name.

newmem

Is the new member name.

REPLACE

Is optional and replaces existing members with the same name as *newmem*.

Since the comma is used in member specifications, each pair of members is separated with one or more blanks when specified in a list. Therefore, a list of member specifications is always enclosed in parentheses.

Example: Using the DYNAM RENAME Command

```
DYNAM RENAME MYID.DATA.OLD MYID.DATA.NEW
DYNAM REN MYID.DATA.LIB MEMBER OLD,NEW,R
DYNAM RENAME MYDD(OLD1,NEW1,R OLD2,NEW2)
DYNAM REN MYDD(OLD1,NEW1 OLD2,NEW2) REPL
```

SUBMIT Subcommand

The DYNAM SUBMIT command submits jobs to z/OS.

Syntax: How to Submit a Job to z/OS

```
DYNAM SUBMIT dname [[MEMBER] members]
```

where:

SUBMIT

Can be abbreviated as SUB.

dname

Is the dsname or ddname of the input data set(s) containing JCL to be submitted. The ddname can specify a concatenation of data sets.

MEMBER

May be omitted if the members are specified in parentheses.

members

May be a single member name or a list of members. When submitting a member list, the resulting job stream is the concatenation of the members. If the members are enclosed in parentheses, blanks before the left parenthesis can be omitted.

Example: Using the DYNAM SUBMIT Command

```
DYNAM SUBMIT MYDD MEMBER ASM,PROG,LKED
DYNAM SUB MYDD(ASM,PROG,LKED)
DYNAM SUB MYID.DATA.LIB(CREATE LOAD)
DYNAM SUBMIT MYFILE
```

Note: The DYNAM SUBMIT command provides an interface with the submit user exit IKJEFF10 as described in the *IBM TSO Extensions Version 2 Customization* manual. For details, see the Information Builders Technical Memo 7859, *Enabling a Site-Specified Submit Exit Routine*, or view FOCCTL.DATA(SUBMITZ2).

COMPRESS Subcommand

The DYNAM COMPRESS command compresses the partitioned data sets (PDS).

Syntax: How to Compress a Partitioned Data Set

```
DYNAM COMPRESS dname [dname]...
```

where:

COMPRESS

Can be abbreviated as COMP.

dname

Is the dsname or ddname of a PDS to be compressed. The ISPF-like lock is obtained.

If the dsname is specified, it is allocated as OLD. If the ddname is specified and status is SHR, you have to make sure that another user does not access the PDS during the compress operation.

Note: DYNAM COMPRESS uses the IBM utility IEBCOPY, and therefore can only be used when running with APF authorization.

Example: Using the DYNAM COMPRESS Command

```
DYNAM COMPRESS MYDD
DYNAM COMPRESS MYID.DATA.LIB
DYNAM COMP MYDD MYID.DATA.LIB
```

Comparison of TSO Commands, JCL, and DYNAM

This section shows examples of TSO ALLOCATE and FREE commands, JCL commands, and the equivalent DYNAM commands.

Example: Allocating an Existing File

TSO:

```
TSO ALLOC F(FOCEXEC) DA('MYUSER.FOCEXEC.DATA') SHR
```

JCL:

```
//FOCEXEC DD DSN=MYUSER.FOCEXEC.DATA,DISP=SHR
```

DYNAM:

```
DYNAM ALLOC FILE FOCEXEC DA MYUSER.FOCEXEC.DATA SHR
```

Example: Creating a New Data Set

TSO:

```
TSO ALLOC F(FOCEXEC) DA('MYUSER.FOCEXEC.DATA') -
    SPACE(5,3) TRACKS CATALOG DIR(2) -
    UNIT(SYSDA) USING(NEWDCB) -
    LRECL(80) RECFM(F B) BLKSIZE(1600)
```

JCL:

```
//FOCEXEC DD DSN=MYUSER.FOCEXEC.DATA,DISP=(NEW,CATLG),
//   SPACE=(TRK,(5,3,2)),UNIT=SYSDA,
//   DCB=(LRECL=80,RECFM=FB,BLKSIZE=1600)
```

DYNAM:

```
DYNAM ALLOC FILE FOCEXEC DA MYUSER.FOCEXEC.DATA -
      SPACE 5,3 TRACKS CATLG DIR 2 UNIT SYSDA -
      LRECL 80 RECFM FB BLKSIZE 1600
```

Example: Freeing Files

TSO:

```
TSO FREE F(FOCEXEC)
```

DYNAM:

```
DYNAM FREE FILE FOCEXEC
```

Example: Concatenating Files

TSO:

```
TSO ALLOC F(FOCEXEC) DA('MYUSER.FOCEXEC.DATA'-
    'MYUSER.PROGRAMS.DATA') SHR
```

JCL:

```
//FOCEXEC DD DSN=MYUSER.FOCEXEC.DATA,DISP=SHR
//   DD DSN=MYUSER.PROGRAMS.DATA,DISP=SHR
```

DYNAM:

```
DYNAM ALLOC FILE FOCEXEC DA MYUSER.FOCEXEC.DATA SHR
DYNAM ALLOC FILE PROGRAMS DA MYUSER.PROGRAMS.DATA SHR
DYNAM CONCAT FILE FOCEXEC PROGRAMS
```

Allocating Temporary Files

Temporary files are transient files that disappear after you end a session.

You can control the size and location of temporary metadata files and data files created by HOLD commands. You can specify that the temporary files reside in the hierarchical file system, MVS data sets, or in hiperspace.

Syntax: How to Allocate Temporary Files

```
DYNAM SET TEMP[ALLOC] {MVS|HIPER}
```

where:

`MVS`
Allocates temporary files to MVS data sets. This is the default value.

`HIPER`
Allocates temporary files to hiperspace.

Note: For z/OS, temporary metadata files can be allocated using a similar procedure to allocating permanent metadata files:

- If DYNAM allocation for HOLDMAST or HOLDACC is present, temporary files are stored in the designated PDSs.
- If DYNAM SET TEMP[ALLOC] MVS is issued, temporary files are stored in the default temporary PDSs.
- If DYNAM SET TEMP[ALLOC] HIPER is issued, temporary files are stored in HIPERSPACE.

Syntax: **How to Allocate Temporary Files to MVS Data Sets**

To alter the default allocation parameters for temporary files for MVS data sets, issue the following command.

```
DYNAM SET TEMP[ALLOC] FOR type dynam_parms
```

where:

type
: Is one of the following: HOLDACC, HOLDMAST, HOLD SAVE, REBUILD, FOCUS, FOCSORT, OFFLINE, or FOC$HOLD.

dynam_parms
: Are regular DYNAM ALLOC parameters to be used as default for that type. Note that DCB parameters, if provided here, will be ignored, since they must be compatible with the file type being written.

  This is similar to the functionality of IBITABLA. The defaults should be overwritten for all cases when a private copy of IBITABLA exists containing different values.

System defaults for HOLDMAST and HOLDACC are:

```
TRKS 5 5 DSORG PO DIR 36 NEW REU
```

System defaults for all other types are:

```
CYLS 5 10 DSORG PS NEW REU
```

Example: **Allocating HOLD Masters**

The following command allocates HOLD Master Files to a data set named USER1.HOLDMAST.DATA

```
DYNAM SET TEMP FOR HOLDMAST DA USER1.HOLDMAST.DATA SHR REU
```

Chapter 7

Using FOCUS as a Client to a Reporting Server

This chapter describes FOCUS client/server computing. An understanding of server components is essential for those planning to implement client/server applications.

In the remainder of this chapter, the term *server* refers to any Reporting Server (WebFOCUS or iWay).

In this chapter:

- Client/Server Computing and Middleware
- Using FOCUS to Access Data on a Server
- Remote Execution
- Distributed Execution

Client/Server Computing and Middleware

Client/server computing enables sites to exploit the power of networked computing systems. Heterogeneous in nature, client/server architecture divides application components such as screen formatting, program logic, and data access between several networked processors to exploit unique characteristics in each environment:

- A client, such as FOCUS for Mainframe, typically builds a request and specifies a report layout.
- A remote server, or back-end, then performs data selection and handles data integration and aggregation.

By enabling numerous front-end tools to share centralized back-end services, client/server computing introduces the concept of *interoperability*. Seamless integration of the front- and back-end processes is accomplished through a new layer of systems software, called *middleware*, which provides interoperability by delivering transparent data transmission and translation services between physically linked processors.

Middleware insulates end users and application developers from dealing with the complexities and incompatibilities of networked proprietary computing environments. This is accomplished through three components:

- **Application Programming Interface (API).** This client component, built into FOCUS for Mainframe, requests remote services using SQL requests.

- **Database Server or Gateway.** This back-end server or gateway translates remote requests into formats suitable for the specified target environment and executes them.
- **Network Communications.** Network communication services provide protocol translation services, masking incompatibilities between proprietary networks and interconnected systems.

The communications system and protocol subsystem, which together make up Network Communications, shield the API and server from details of various communications protocol syntaxes. Each protocol subsystem is a platform specific interface to a supported communications protocol. Together, these components enable applications to send requests to a variety of servers and to receive answers (data or messages) in return.

Using FOCUS to Access Data on a Server

This topic describes processing alternatives when using FOCUS to access data on a server:

Remote Execution	This approach allows you to read, analyze, consolidate, and update remote data by sending the FOCUS stack to a server. You can also execute FOCEXECs stored on the server (called Remote Procedures), using a process known as a Remote Procedure Call (RPC). In remote execution, the server typically handles data processing, while the client does the request generation and data presentation. To return report output to a file on the FOCUS Client, you can use the PCHOLD (HOLD AT CLIENT) command in your report request.
Distributed Execution	In Distributed Execution (also referred to as the Server Data Adapter), data retrieved from the server is processed on the client. Here, the server is primarily involved with data access. Distributed Execution shields end users from needing to know where data actually resides through a mechanism known as location transparency.

When you use FOCUS as a client to a Reporting Server, you can control whether the:

- Source code for your procedures appears in the batch output.

- USAGE formats in HOLD Master Files are taken from the original Master File or determined by the attributes of the output returned by the request.

Syntax: How to Prevent Source Code From Displaying in Batch Output

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

where:

`ON`

Suppresses FOCEXEC source code from displaying in the batch output.

`OFF`

Displays the code. This is the default value.

Syntax: How to Control USAGE Attributes in a HOLD Master File

```
SQL EDA SET USAGEFORMAT {ON|OFF}
```

where:

`ON`

USAGE formats and edit options in the HOLD Master File match those in the original Master File.

`OFF`

Causes USAGE formats of HOLD Master Files to be defined based on the data string that is returned by the Server API.

For example, a field described as P8.2 in the original Master File is described as P10.2 in the HOLD Master File. The additional bytes account for the decimal point as well as a minus sign in the case of a negative value. This is the default value.

Establishing and Configuring the FOCUS User Environment

Startup FOCEXECs can log FOCUS users onto a remote server and establish environmental conditions for a session. PROFILE FOCEXECs can also establish environmental conditions or build menu shells of user options.

On the server, the PROFILE FOCEXEC can establish the working environment for the Reporting Server for z/OS—for example, setting WIDTH and PANEL parameters for reporting.

To use a Reporting Server with FOCUS for Mainframe, you must have the following products installed:

- Any current release of a server on any supported platform.
- FOCUS for Mainframe, to use as the client. Depending on your method of remote execution (described in *Remote Execution* on page 239 and *Distributed Execution* on page 246), you also need a server configuration file.

Server Configuration File

You must have a configuration file allocated to ddname EDACS3, EDACFG (or CONFIG) to use either remote execution or distributed execution (distributed execution is also called SUFFIX=EDA). It is possible to switch back and forth between these execution techniques in your session.

DNS Names Support

The FOCUS Client configuration file can identify a server by host name or IP address.

The Domain Name System (DNS) is a global network of servers that translate host names, such as www.informationbuilders.com, into IP addresses.

Syntax: How to Invoke DNS Names Support

The syntax for specifying a host name in the client configuration file for TCP/IP is

```
HOST = hostname
```

where:

hostname

Is the name of the host where the server resides.

Example: Using DNS Names Support

The following client configuration file is used for connecting to the host named IBIMVS:

```
NAME = EDA CLIENT USING CS/3 TCP/IP
NODE = TCPOUT
  BEGIN
 ;  TRACE = 31
    PROTOCOL = TCP
    CLASS = CLIENT
    HOST = IBIMVS        ; DNS NAME OF HOST
    SERVICE = 2459       ; PORT NUMBER OF SERVER
  END
```

Remote Execution

Remote execution allows users to transmit local FOCUS requests to a remote host for execution. In this operating mode, a request built on the client is shipped to the server for execution. On completion, the server returns the output to the client, which displays it in Hot Screen. The server does all data processing, while the client handles request generation and data presentation. Remote execution also supports server-based requests (Remote Procedures), through Remote Procedure Calls (RPCs).

Before sending a request to a remote server, you must connect to it by either:

- Issuing the REMOTE DESTINATION, REMOTE USERID, and REMOTE PASSWORD commands at the FOCUS prompt.

  or

- Creating a FOCEXEC that issues the REMOTE DESTINATION, REMOTE USERID, and REMOTE PASSWORD commands for you.

Note: All data sources supported by servers can be accessed using remote execution.

Logging On With REMOTE Commands

To have FOCUS automatically log you on to a server, create a FOCEXEC containing the REMOTE DESTINATION, REMOTE USERID, and REMOTE PASSWORD commands.

- The REMOTE DESTINATION command directs requests to a particular server.
- The REMOTE USERID command sets the user ID for logging on to the server. When FOCUS issues a request, it sends the user ID to the security system on the server (for example, RACF on z/OS).
- The REMOTE PASSWORD command sets the user password. When FOCUS issues a request, it sends the password to the security system on the server (for example, RACF).

Syntax: How to Specify a Target Server: REMOTE DESTINATION Command

```
REMOTE DEST[INATION] = server
```

where:

server

Is the server name from the client configuration file. It must match the SERVICE attribute in the server configuration file (allocated to the EDASERVE ddname).

Example: Issuing the REMOTE DESTINATION Command

The following server configuration file contains the attribute SERVICE=IBIEDA:

```
*****************************************************************
**                SERVICE for CS/3 Communications              **
*****************************************************************
SERVICE      = IBIEDA
PROGRAM      = TSCOM3
NUMBER_READY = 0
MAXIMUM      = 50
*IDLELIM      = 20
SERVINIT     = *,++
  AUTOSTART  = NO
...
```

The client configuration file that provides access to this server contains the attribute NODE=IBIEDA:

```
NODE = IBIEDA
 BEGIN
;  TRACE = 31
   PROTOCOL = TCP
   CLASS = CLIENT
   HOST = IBIMVS       ; DNS NAME OF HOST
   SERVICE = 2386      ; TCP/IP PORT THAT SERVER IS LISTENING ON
END
```

To connect to the server in a FOCEXEC, issue the following command:

```
REMOTE DEST = IBIEDA
```

Syntax: How to Identify the User: REMOTE USERID Command

```
REMOTE USERID = serveruserid
```

where:

serveruserid

Is your user ID.

This user ID remains in effect until you reissue the REMOTE USERID command.

Syntax: How to Authenticate the User: REMOTE PASSWORD (USERPASS)

```
REMOTE PASSWORD = serverpassword
```

where:

serverpassword

Is your password.

This password remains in effect until you reissue REMOTE PASSWORD.

Note: By not specifying the REMOTE USERID and/or REMOTE PASSWORD commands, the security ID and password used for your mainframe session (z/OS user ID) are used to sign on onto the server.

Sending Requests to a Remote Server

You can send FOCUS requests to the server for execution as follows:

- Use the REMOTE EX command from the FOCUS Session prompt to name a FOCEXEC on the client that you want to execute on the server.
- Include the commands -REMOTE BEGIN and -REMOTE END around the code you are sending to the server. Then execute the procedure as you would any other FOCUS procedure. Expansion of Dialogue Manager amper variables takes place *on the client*, before the request is forwarded to the server.
- Use TableTalk to create the request and select the *Execute on Remote Server* option on the final TableTalk screen.

When the server returns report output, FOCUS displays the report in Hot Screen.

Syntax: How to Execute a Request Remotely Using the REMOTE EX Command

Issue the following command at the FOCUS Session prompt:

```
REMOTE EX focexecname
```

where:

focexecname

Is the name of a FOCEXEC stored on the client that is to be executed on the server. Note that Dialogue Manager amper variables in the FOCEXEC will be expanded on the client before the request is shipped to the server.

Important: A request executed this way may not contain the commands -REMOTE BEGIN and -REMOTE END. REMOTE EX issues those commands automatically.

Example: Executing a Request Remotely Using the REMOTE EX Command

The following sample FOCEXEC (EDHOURS) is executed at the FOCUS Session prompt with the REMOTE EX command.

The following code is stored on the client as FOCEXEC EDHOURS:

```
TABLE FILE EMPLOYEE
  HEADING CENTER
  "SUMMARY REPORT OF EMPLOYEE CLASSROOM HOURS"
  " "
  SUM ED_HRS BY EMP_ID
END
```

To execute this procedure on the server, issue the following command:

```
>> REMOTE EX EDHOURS
```

Using -REMOTE BEGIN and -REMOTE END to Execute Requests Remotely

Another way to execute report requests against remote data is to begin the FOCEXEC that you want executed on the server with the command -REMOTE BEGIN, and follow the END command with -REMOTE END. Then execute the FOCEXEC.

Using -REMOTE BEGIN and -REMOTE END provides the following advantages:

- The command for executing the procedure is the same that you would use locally.
- By adding -REMOTE BEGIN and -REMOTE END, developers can separate FOCEXECs into different components, each of which may be executed remotely or locally.
- You do not need to have a Master File for the target data source on the client with this approach.

Keep the following restrictions in mind:

- When developing applications (that is, when coding FOCEXECs), you can use multiple -REMOTE BEGIN and -REMOTE END pairs in the FOCEXEC. You cannot nest -REMOTE BEGIN and -REMOTE END pairs.
- If a -RUN command falls between a -REMOTE BEGIN and -REMOTE END command pair, it is treated as if it were a -REMOTE END, followed immediately by a -REMOTE BEGIN. All FOCUS commands preceding the -RUN are sent up to and executed on the server. Commands following the -RUN are placed on the FOCUS stack when control returns to the client and are executed on the server when the -REMOTE END is encountered.

- You cannot mix REMOTE commands in a FOCEXEC. For example, you cannot use the REMOTE EX command for a FOCEXEC that contains -REMOTE BEGIN and -REMOTE END commands. This restriction applies to other FOCEXECs executed from within your main FOCEXEC by -INCLUDE or EX commands.

Example: Executing Requests Using -REMOTE BEGIN and -REMOTE END

The following example demonstrates the use of a Dialogue Manager amper variable (&1) with FOCUS commands in a FOCEXEC named MARGIN. The request executes a JOIN and report request on the server. The numbers on the left refer to the notes that follow.

```
1. -REMOTE BEGIN
2. JOIN PROD_CODE IN &1 TO PROD_CODE IN PROD AS AJOIN
   TABLE FILE &1
   PRINT UNIT_SALES
   AND COMPUTE
   MARGIN/D8.2= RETAIL_PRICE - UNIT_COST;
   BY STORE_CODE BY PROD_CODE
   END
3. -REMOTE END
```

1. -REMOTE BEGIN identifies the beginning of the command stream to be executed on the server. Any commands already in the FOCUS stack are executed when -REMOTE BEGIN is encountered.
2. The MARGIN procedure includes FOCUS commands and an amper variable, which is expanded on the client when used with -REMOTE.
3. -REMOTE END identifies the end of the command stream to be executed on the server.

Execute MARGIN and provide the name of the target data source as an argument on the command line. For example, to obtain the margin report for the SALES data source, issue the following command at the FOCUS prompt:

```
EX MARGIN SALES
```

SALES is substituted for the Dialogue Manager variable &1 in the JOIN and TABLE commands, and the commands then execute on the server. The resulting report is returned by the server and displayed in Hot Screen on your terminal.

Syntax: How to Execute Remote Procedures in Remote Execution Mode

In addition to making data available to the client, a server can also hold remote or stored procedures that can be executed from FOCUS.

To execute a procedure that resides on the server, use the following syntax:

```
>> REMOTE EX focexec
```

where:

focexec

Is a FOCEXEC on the client that contains a command that executes a procedure on the server.

Example: Executing a Remote Procedure in Remote Execution Mode

FOCEXECL resides on the client, and FOCEXECS resides on the server.

FOCEXECL contains the following command:

```
EX FOCEXECS
```

To execute FOCEXECS, issue the following command:

```
REMOTE EX FOCEXECL
```

Another way to do this is to issue the following commands:

```
-REMOTE BEGIN
EX FOCEXECS
-REMOTE END
```

Example: Using -INCLUDE to Call a FOCEXEC Stored on the Client

You can use the -INCLUDE command in a FOCEXEC to call another FOCEXEC stored on the client.

This example downloads data from a host to a FOCUS Client and updates the EMP data source.

```
-REMOTE BEGIN
TABLE FILE EMPLOYEE
 PRINT EMP_ID SALARY START_DATE
 ON TABLE HOLD AT CLIENT AS LD
END
-REMOTE END
-INCLUDE FOCEXECL
```

FOCEXECL contains the following commands:

```
MODIFY FILE EMP
 FIXFORM FROM LD
 DATA ON LD
END
```

Note: If you use remote execution to execute a FOCEXEC on the server that uses a -INCLUDE command, the FOCEXEC named must reside on the server and the -INCLUDE command must precede the -REMOTE END command.

Viewing a System or Error Message

All FOCUS and system messages returned by the server are displayed, as are error messages resulting from remote execution.

FOCUS places the error message numbers in the Dialogue Manager variable &FOCERRNUM, which can then be tested in FOCEXECs.

Syntax: How to Display a System or Error Message

```
? nnnn
```

where:

nnnn

Is the error message number.

Terminating the Remote Session: REMOTE FIN

The REMOTE FIN command logically terminates a FOCUS session with a server. It should be issued at the conclusion of remote data access. The server must be available when you issue this command.

Note: Issuing a FIN command in native FOCUS closes all active sessions.

Syntax: How to Terminate a Remote Session With the REMOTE FIN Command

```
REMOTE FIN server
```

where:

server

Must match the server name in the configuration file.

Querying Remote Session Parameter Settings: ? REMOTE

The ? REMOTE command displays all remote session parameters in effect. Issue it at any time in your FOCUS session.

Syntax: How to Query Remote Session Parameter Settings

```
? REMOTE
```

The output is:

```
Remote Destination   --> IBIEDA
Remote User ID       --> USER1
Remote User Password --> ........
Conversations exist with :
  IBIEDA (EDA5.2)
```

Distributed Execution

With distributed execution (also known as the Server Data Adapter or SUFFIX=EDA), you can access all data sources accessible to a server. Your Master Files and Access Files tell FOCUS where to find the data.

The Server Data Adapter provides the following advantages:

- Once you set up your Master Files and Access Files on the client, you can use FOCUS to access remote data just as if it were local data.
- You can join data sources across platforms. For example, join a data source on z/OS to a data source on a UNIX platform.

Syntax: How to Implement Distributed Execution

Using the following SUFFIX attribute in a Master File directs FOCUS to pass all requests for that data source directly to the data adapter, which passes them on to a server:

```
SUFFIX=EDA
```

You can establish the name of the target server in either of the following ways:

- Store the name of the target server in an Access File. The syntax in the Access File is

  ```
  SERVER = servername
  ```

  where:

  servername

  Is the name of the target server (value of the NODE attribute in the client configuration file).

  The ddname of the Access File is FOCSQL.

- Issue the following command:

  ```
  SQL EDA SET SERVER servername
  ```

A server name in an Access File overrides any name specified in an SQL EDA SET SERVER command. By removing the SERVER attribute from the Access File, you can dynamically control the server location with SQL EDA SET SERVER commands.

Your Access File member names must match your Master File member names and must reside in a partitioned data set allocated to ddname FOCSQL. For example,

```
DYNAM ALLOC F1 MASTER DS userid.MASTER.DATA SHR REU
DYNAM ALLOC F1 FOCSQL DS userid.FOCSQL.DATA SHR REU
```

where:

userid.MASTER.DATA(CAR)

Is the CAR Master File.

userid.FOCSQL.DATA(CAR)

Is the CAR Access File.

Example: Storing a Server Name in an Access File

The following example shows how to store server name IBMSERVE in an Access File.

```
SEGNAME=ONE,TABLENAME=CAR,KEYS=1,WRITE=YES,SERVER=IBMSERVE,$
```

Note: Regardless of the type of data source to be accessed, the Access File must contain the following attributes:

- SEGNAME. The segment name in the Access File must match the segment name in the Master File.
- TABLENAME. This attribute specifies the name of the Master File on the server.
- SERVER. This attribute specifies the name of the server.

Example: Submitting a Request Using SUFFIX=EDA

Consider the following request:

```
TABLE FILE DIGITEDA
PRINT *
END
```

The Master File named DIGITEDA on the client is:

```
FILENAME=DIGITEDA,SUFFIX=EDA,$
 SEGNAME=DIGIT,SEGTYPE=S0,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I6   ,I4             ,$
  FIELD=SSN        ,SSN        ,A9   ,A9 ,MISSING=OFF,$
  FIELD=AMOUNT1    ,AMOUNT1    ,P8   ,P8 ,MISSING=ON ,$
  FIELD=AMOUNT2    ,AMOUNT2    ,P9.0 ,P8 ,MISSING=ON ,$
```

The Access File named DIGITEDA on the client is:

```
SEGNAME=DIGIT,TABLENAME=DIGIT,KEYS=1,SERVER=PMSEDA,$
```

The Master File (named DIGIT) on the server is:

```
FILENAME=DIGIT,SUFFIX=FOC,$
 SEGNAME=DIGIT,SEGTYPE=S0,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I9    ,I4 ,MISSING=ON,$
  FIELD=SSN        ,SSN        ,A9    ,A9 ,MISSING=ON,$
  FIELD=AMOUNT1    ,AMOUNT1    ,P16.0 ,P8 ,MISSING=ON,$
  FIELD=AMOUNT2    ,AMOUNT2    ,P16.0 ,P8 ,MISSING=ON,$
```

The EDACS3 client communication configuration file is:

```
NAME = EDA CLIENT USING CS/3 TCP/IP
NODE = PMSEDA
 BEGIN
;  TRACE = 31
   PROTOCOL = TCP
   CLASS = CLIENT
   HOST = IBIMVS           ; DNS NAME (PNO 28109)
   SERVICE = 2386          ; TCP/IP PORT FOR SERVER
END
```

The TABLE request references a local Master File named DIGITEDA. Its corresponding Access File contains information such as the server name and the Master File name as it is known at the server. In this case, Server PMSEDA contains a Master File called DIGIT. At the server, the DIGIT Master File describes a FOCUS data source. The communications configuration file contains an entry for server name PMSEDA so that the FOCUS Client can establish communications with the server.

Similarly, SQL can be used to reference the Master File. For example:

```
SQL SELECT * FROM DIGITEDA
END
```

Example: Using a Remote Multi-Segment Master and Access File

The following is a multi-segment Master File:

```
FILENAME=JOINEDA, SUFFIX=EDA
 SEGNAME=EMPDATSE, SEGTYPE=S0
  FIELDNAME=EMP_ID,     ALIAS=EMP_ID,     FORMAT=A9,  INDEX=I, $
  FIELDNAME=LAST_NAME,  ALIAS=LAST_NAME,  FORMAT=A15,          $
  FIELDNAME=FIRSTNAME,  ALIAS=FIRSTNAME,  FORMAT=A10,          $
  FIELDNAME=MIDINITIAL, ALIAS=MIDINITIAL, FORMAT=A1,           $
  FIELDNAME=DIV,        ALIAS=DIV,        FORMAT=A4,           $
...
 SEGNAME=DIGIT,SEGTYPE=S0,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I4    ,I4 ,MISSING=OFF,$
  FIELD=THIS_DIGIT ,THIS_DIGIT ,I9    ,I4 ,MISSING=OFF,$
  FIELD=SSN        ,SSN        ,A9    ,A9 ,MISSING=OFF,$
  FIELD=AMOUNT1    ,AMOUNT1    ,P16.0 ,P8 ,MISSING=ON ,$
  FIELD=AMOUNT2    ,AMOUNT2    ,P16.0 ,P8 ,MISSING=ON ,$
```

The following is the corresponding multi-segment Access File:

```
SEGNAME=EMPDATSE, TABLENAME=EMPLOYEE, KEYS=0 ,SERVER=PMSEDA,$
SEGNAME=DIGIT   , TABLENAME=DIGIT   , KEYS=0 ,SERVER=PMSEDA,
  KEYFLD=EMP_ID, IXFLD=SSN ,$
```

How Location Transparency Works

Distributed execution provides location transparency because, once a Master File and Access File have been generated, end users can access data in the data source without knowing where it resides. You can manipulate data sources described with SUFFIX=EDA as if they were local. The data returned by a server can be converted to any format supported by FOCUS.

Generally, you create Master Files and Access Files only once. It is necessary to regenerate them only if the structure of the data source on the server changes.

Keep the following in mind:

- FOCUS can join data sources that reside on different platforms. The communication protocols used to connect to the servers may be the same or different (for example, TCP/IP to a Server for VMS, and LU0 to a Server for z/OS).
- The Server Data Adapter (SUFFIX=EDA) is a relational view. Like other relational data adapters such as Oracle or SQL Server, it uses the ALIAS value from a Master File when generating a report request to send to the DBMS. Consequently, make sure that you provide values for the ALIAS attributes in your data source descriptions.
- Master Files should also contain values for ACTUAL format attributes.

Logging On to the Server With Distributed Execution

You can log on to a server by issuing SQL EDA SET commands in a FOCEXEC or at the FOCUS Session prompt.

- The SQL EDA SET SERVER command sets the server destination (an alternative to placing the server name in the Access File).
- The SQL EDA SET USER command sends a user ID and password to the server.

Syntax: How to Set the Server Destination With Distributed Execution

```
SQL EDA SET SERVER servername
```

where:

servername

Is the server name. It must match the SERVICE keyword in the configuration file on the server.

Syntax: How to Send a User ID and Password to the Server With Distributed Execution

```
SQL EDA SET USER servername/userid,password
```

where:

servername

Is the server with which you want to associate this user ID and password.

userid

Is the user ID.

password

Is the password.

This command does not determine the server to which requests will be directed. It simply associates a user ID and a password with a particular server.

Joining Data Sources Across Platforms With Distributed Execution

You can use the Server Data Adapter to join data sources on different platforms. The presence of SUFFIX=EDA in the Master File causes FOCUS to use the data adapter. For example, you can join an EMPLOYEE data source on a z/OS system to a JOBFILE data source on a UNIX system. The data adapter works faster if you join the smaller data source to the larger data source; the data adapter makes one SQL call to the first data source, then makes one SQL call to the second data source, for each value of the referenced field.

Issuing SQL Commands to the Server With Distributed Execution

You can issue FOCUS or SQL commands to the server. Issue the SQL commands at the FOCUS Session prompt or place them in FOCUS procedures. For example, from the FOCUS Session prompt, you could issue the following commands:

```
> sql eda
> select * from car
> end
```

Executing Stored Procedures With Distributed Execution

In addition to making data available to the client, a server can store procedures, which are called remote procedures or stored procedures.

Syntax: How to Execute a Stored Procedure Using Distributed Execution

You can execute stored procedures from FOCUS, by issuing the command:

```
SQL EDA EX rpcname parm1, parm2, ...
END
```

where:

rpcname

Is a procedure on the server.

parm1, parm2,...

Are character strings sent to the server (they are the same as parameters you can pass on the execution line of a FOCEXEC).

Syntax: How to Query Server Data Adapter Settings

To view Server Data Adapter parameter settings, issue the command:

```
SQL EDA ?
>
```

The output is:

```
> sql eda ?
(FOC1450) CURRENT EDA INTERFACE SETTINGS ARE :
(FOC1446) DEFAULT DBSPACE IS                    -  : IBIEDA
(FOC1449) CURRENT SQLID IS                      -  : USER1
(FOC1444) AUTOCLOSE OPTION IS                   -  : ON FIN
(FOC1496) AUTODISCONNECT OPTION IS              -  : ON FIN
(FOC1499) AUTOCOMMIT OPTION IS                  -  : ON COMMAND
(FOC1441) WRITE FUNCTIONALITY IS                -  : OFF
(FOC1445) OPTIMIZATION OPTION IS                -  : ON
(FOC1484) SQL ERROR MESSAGE TYPE IS             -  : DBMS
(FOC1552) INTERFACE DEFAULT DATE TYPE           -  : NEW
```

Using SQL Passthru With Distributed Execution

The Server Data Adapter provides an SQL Passthru mode, through which SQL requests can be sent directly to the server and passed to a relational DBMS with no translation by FOCUS. To use SQL Passthru:

1. Invoke SQL Passthru mode on the server through the use of a server profile or stored procedure.

   For example, if you want to use SQL Passthru with DB2, you would execute the command:

   ```
   SQL EDA SET ENGINE DB2
   ```

   For more information, see your Reporting Server documentation.

2. Execute an SQL Passthru command either from the FOCUS Session prompt or in a FOCUS application using the format

   ```
   SQL EDA sqlstatement
   ```

   where:

   sqlstatement

   Is a valid SQL statement.

You do not need a Master File or Access File on the client when using SQL Passthru.

Chapter 8

Logging FOCUS Usage: FOCLOG

FOCLOG is a tool for recording and analyzing the use of FOCUS for your entire site. It comes packaged with a set of standard analytical reports that allow you to interrogate FOCUS usage—identify usage spikes and redundancies, detect large report requests, analyze time-of-day usage trends, and monitor ad hoc versus scheduled requests for each user. It even allows you to analyze the environmental conditions of the query, such as use of joins, cross references, combines, MSO, or SU. In addition, it collects and reports on statistics, such as the number of data rows extracted and number of lines on the report output.

FOCLOG is invisible and non-intrusive to your production applications, whether batch or online. Rolled up and sorted in creative ways, the captured data provides the insight a site coordinator or manager needs to gain a clear picture of FOCUS usage and to target areas that could require adjustment, consolidation, or expansion.

In this chapter:

- Overview of FOCLOG
- Implementing FOCLOG
- Information Captured in the FOCLOG File
- FOCLOG Reporting

Overview of FOCLOG

FOCLOG is a facility for logging FOCUS usage that was designed to have a negligible impact on FOCUS applications and to make it easy to analyze the collected data.

The cost of logging has been made so low as to be insignificant, so FOCUS usage can now be monitored continually, not just for selected periods. This is because the log is a simple sequential file to which usage data gathered during the FOCUS session or batch job is only appended when the FOCUS session ends.

Updates to the log from concurrently executing FOCUS sessions or batch jobs are serialized through a standard systems-wide z/OS Enqueue macro. The systems-wide scope allows a single log to gather usage data from users in different LPARS on the same machine.

The log is a physical sequential file with multiple record types that is easily read by FOCUS using a distributed Master File. Multiple logs can be logically concatenated so they are viewed as a single entity, and analyzed right where they are. Alternatively, since all the data is in character format, it can easily be moved to another platform—for example a laptop—where a more graphical analysis can be done using WebFOCUS.

How Logging Is Implemented

The FOCLOG facility is incorporated into FOCUS. However, logging is only triggered if at the start of execution:

- The command SET FOCLOG=ON is in effect. You can add this command to the FOCPARM or FOCPROF profile (member FOCPARM or FOCPROF in the concatenation of data sets allocated to DDNAME ERRORS).
- FOCUS detects the presence of member FOCUSLOG in any of the data sets allocated to DDNAME ERRORS. This file in turn names the log data set itself. The ERRORS DDNAME is already allocated through JCL in all production FOCUS environments—FOCUS will not execute without it.

To inhibit logging, you can delete or rename the FOCUSLOG file or set the FOCLOG parameter to OFF.

When logging is triggered, usage data is collected and kept in memory as the application proceeds. The log itself is not opened until the FOCUS session or batch job ends. If FOCUS abends in the course of execution, or if it is stopped by the operator, the usage data will not be logged for those user sessions active at the time of the stoppage.

Extreme precautions have been taken to ensure that logging difficulties will not affect the application. If the log cannot be written for whatever reason (for example, it is full, the user has no write privilege to it, there has been an I/O error), the only effect on the application will be that the usage data will not be logged. This is true even if the logging difficulty results in an abend. FOCUS will recover and terminate normally.

Failures to log are not signaled through FOCUS-generated error messages.

If the logging failure caused an abend from which FOCUS recovered—for example a B37 abend caused by a log full condition—the occurrence of the abend will be recorded in the batch job step statistics or in your TSO log, but logging failures which do not cause an abend will not be recorded in any way.

The z/OS ISPF utility can be used to examine the date and time of the last update of the log.

The Log Data Set

The first 44 bytes of the first record of the FOCUSLOG member in the concatenation of data sets allocated to DDNAME ERRORS contains the fully qualified data set name of the current log file.

Other records in the FOCUSLOG file are ignored, so they can safely be used to retain the names of prior logs. The default is to create short logs; long logs are created by following the log file name with the keyword DETAIL. This keyword gives you everything recorded in the short log and adds the data set or file names for the FOCEXEC, Master File, and Access File used in each request.

The site administrator must ensure that the LOG file is write-accessible to all FOCUS users and protected against archiving—FOCUS will not wait until an archived log is restored, rather it will skip logging instead. Therefore, the log file should *not* be managed by DFSMS.

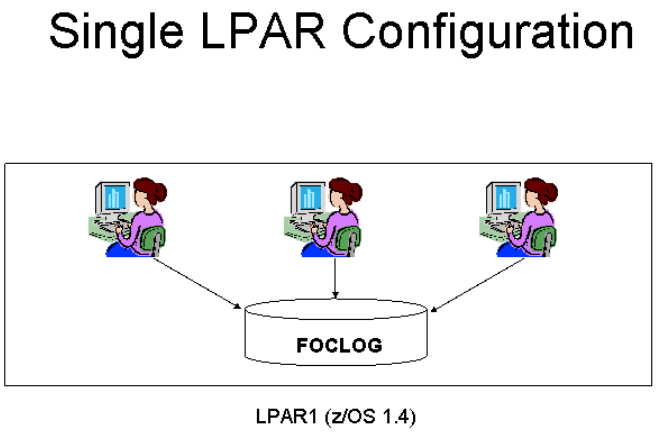
The log consists of four types of records:

Type of record	Size in bytes	Number of Occurrences
Session	168	One per FOCUS session or batch job
Command	129 or 173	One per FOCUS command that accesses data
File	28 or 116	One per MASTER file referenced by the command
Dataset	68	One per data set still allocated at the end of the job

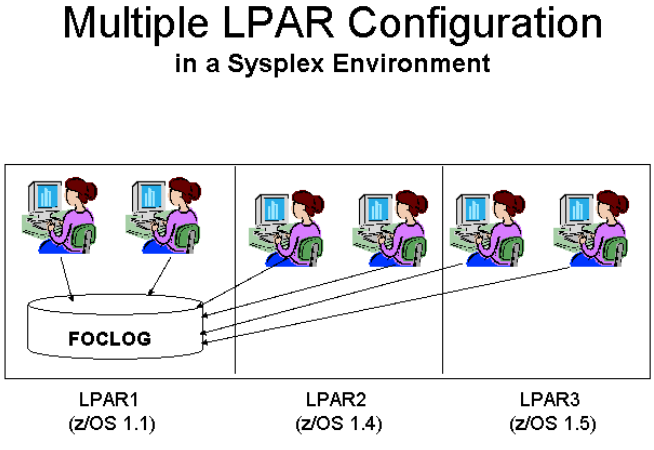
The detailed description of each record is in the distributed FOCLOG Master File and in *Information Captured in the FOCLOG File* on page 265. Long and short logs have identical Master Files; in the short log the Master File, Access File, and FOCEXEC data set names will appear blank.

Sample FOCLOG Configuration Scenarios

In the following diagram of a single LPAR configuration, one FOCLOG file records the usage of all FOCUS users in that LPAR.



In a site configuration where FOCUS resides in several LPARS, even with different OS levels, a single FOCLOG file can reside in any one of the LPARs.



Implementing FOCLOG

The person implementing FOCLOG should be a system administrator with the ability to create and allocate data sets and who can assign RACF rules. Knowledge of FOCUS is not required.

The modules, FOCEXECs, and Master File needed to run FOCLOG and the FOCLOG reports are distributed as part of the standard FOCUS libraries. No installation steps are required. The administrator only needs to:

1. Issue the command SET FOCLOG=ON in the FOCPARM or FOCPROF profile.
2. Create the FOCUSLOG file.

   Create a member named FOCUSLOG in the ERRORS.DATA library.
3. Create a log file and activate it by specifying its fully qualified name (data set name) in the FOCUSLOG file.

If your site standards require you to implement new features in a test environment, create a temporary FOCUSLOG ERRORS file.

Create a copy of your FOCUS production CLIST and concatenate your ERRORS library in front of the FOCUS production ERRORS library.

Create a temporary log file and place its fully qualified SFS directory name in your private version of the FOCUSLOG ERRORS file.

After validating the FOCLOG implementation, remove your private FOCUSLOG file from your CLIST or minidisk.

Overview of FOCLOG Implementation

These steps provide a high-level overview of the FOCLOG implementation process:

1. Allocate the FOCLOG data set.
2. Add the SET FOCLOG=ON command to the FOCPARM or FOCPROF profile.
3. Activate FOCLOG.
4. Validate the FOCLOG configuration.
5. Move FOCLOG to your production FOCUS environment.
6. Run the reports supplied with FOCLOG after collecting log data for whatever period you deem appropriate.

Reference: Usage Notes for FOCLOG

- ❑ To avoid archiving the FOCLOG data set, it should not be managed by DFSMS.

- To keep the FOCLOG file a manageable size, the log was designed to capture only those commands that allow you to analyze the function of FOCUS applications at your site. Therefore, by design, only the following commands are captured in the log file: RUN (to run a compiled MODIFY), MAINTAIN, CREATE, FSCAN (as FOC$SCAN), GRAPH, HOLD, MATCH FILE, MODIFY, RETYPE, SCAN, TABLE, and TABLEF. Other commands issued during one of these requests (for example, JOIN or CHECK FILE) are not captured in the log file. However, if a JOIN or cross reference is in effect, all of the joined files are listed in the log after a TABLE request that references the join. MATCH FILE captures only the first file.
- The following commands are not captured in the log file: REBUILD, JOIN, CHECK FILE, MORE.
- In the event that the log file becomes full, recording stops for all users. To replace the log file, you can rename the full log file and allocate another empty file with the name of the original log file. Logging will resume the next time a user session ends (at FIN).

Allocating the FOCLOG Log File

You must have a log file allocated and reference it in the FOCUSLOG ERRORS file in order to initiate logging. You should first determine the size of the log file required as described in *How to Estimate the Size of the Log File* on page 259.

Procedure: How to Allocate the FOCLOG Log File

1. Allocate a physical sequential file with LRECL 256 and RECFM VB using the size estimate you determined in Step 1.

   Edit and submit the following job to allocate the FOCLOG log file after adding an appropriate job card:

   ```
   //* add job card here
   //FLALLOC  EXEC PGM=IEFBR14
   //FOCLOGF  DD DSN=flhlq.FOCLOG.DATA,DISP=(NEW,CATLG),
   //            VOL=SER=volser,UNIT=unit,SPACE=(CYL,(p,s)),
   //            DCB=(RECFM=VB,LRECL=256,BLKSIZE=12288),DSORG=PS
   /*
   ```

   where:

   flhlq

   Is the high-level qualifier for the FOCLOG file.

   unit

   Is a valid DASD unit for the FOCLOG file.

volser

Is the volume serial number of the unit on which you want to store the FOCLOG file.

p

Is the primary space allocation in cylinders.

s

Is the secondary space allocation in cylinders.

2. Using your security interface (for example, RACF), authorize all of your FOCUS users in the entire sysplex to have write access across the sysplex to this file.

The following are suggestions for naming conventions for your log file:

- Create a separate log for each day of the week, for example FOCLOGMO, FOCLOGTU, FOCLOGWE, FOCLOGTH, FOCLOGFR, FOCLOGSA, and FOCLOGTU.
- Use a numbering convention such as FOCLOG1, FOCLOG2, FOCLOG3, and so on, to name multiple log files.
- Name log files by system and date, such as S1091507. Where S1 represents System number 1 and 091507 represents the day of the year (in this example, September 15, 2007).
- Allocate the log file as a Generation Data Group (GDG) in which case the system creates a naming convention by using index number for each generation.

Procedure: How to Estimate the Size of the Log File

The size of the log file required depends on factors, such as the number of users and FOCUS applications in your environment, plus the length of time for which you wish to capture information.

Consider these suggestions when estimating its size:

Estimating user session statistics. A sample user session lasting one hour and containing 26 DD allocations running 25 TABLE requests will produce approximately 77 output records to the log. There will be one session record for the user session, 26 DSNS records (one per DD allocation), and 50 records from the TABLE requests (2 per request). Assuming that the average number of bytes per record is 50, this sample user session will produce 3850 bytes of output in the log.

One cylinder on an IBM 3390-3 device contains 737280 bytes. Therefore, approximately 190 user sessions of this size would fill one cylinder (737280 / 3850).

To apply this calculation at your site, you must evaluate the number of allocations in your production FOCUS CLIST/JCL and estimate how many TABLE requests each session is likely to run in an hour. Then use the following formulas to estimate the size of the log file:

```
bytes_per_hr = no_of_users * ((1+ number_of_allocs + (2 *
avg_no_of_TABLEs)) * 50)

user_sessions_per_cylinder =  737280  / bytes_per_hr
```

Determine how many user sessions you wish to log per day, week, or month and allocate cylinders accordingly. For example, using the sample session described above, eight cylinders would be adequate for collecting 190 user sessions per hour for an eight hour period.

Collecting sample data. If you do not know the level of FOCUS usage at your site, and cannot apply the above technique, you may just wish to collect data for a given period using a large log file and see how long it takes to fill or how much is used. For example, allocate a 100-cylinder log file for use in production for one business day, and at the end of the day, use ISPF to determine how much of the log was populated. If you used 10 cylinders of the log file on a fairly typical day, a good estimate for the size is 10 cylinders per day. Extend this to determine numbers of cylinders required for longer periods. The only penalty for under-sizing your log is that logging stops when the log is full.

In designing the log file, also take the following considerations into account:

- You can easily FTP or email small logs to other locations (for example, your Desktop) for analysis with tools such as WebFOCUS or Excel.
- Large log files can capture data for longer periods without continuous monitoring, but require more time for usage review and are slower when generating reports.

Activating FOCLOG

Usage can be logged at three levels, producing either a short form log, a detailed log, or a summary log, as described in the examples that follow.

Activation of logging requires four components:

1. The command SET FOCLOG=ON must be in effect prior to running procedures whose usage you want captured in the log. You can set this parameter in the FOCPARM or FOCPROF profile to ensure FOCLOG is activated for the FOCUS session.
2. The presence of a FOCUSLOG member in the concatenation of data sets allocated to DDNAME ERRORS.
3. The FOCUSLOG file must contain the name of the FOCLOG log file.
4. An indicator for the type of log desired.

Note: If you perform these steps, you will automatically activate FOCLOG. Do not perform these steps until you are ready to activate FOCLOG.

Example: Requesting a Default Form of the Log

The FOCUSLOG file in this example points to a FOCLOG file named FLHLQ.FOCLOG.DATA. Because the data set name is followed by blanks, this generates a default form of the log containing information about all attributes defined in the FOCLOG Master File, except the data set names of the FOCEXEC files, Master Files, and Access Files.

```
FLHLQ.FOCLOG.DATA
```

Example: Requesting a Detailed Form of the Log

This FOCUSLOG member produces a detailed form of the log containing everything in the FOCLOG Master File plus the data set names of FOCEXEC files, Master Files, and Access Files.

To create the detailed log, the FOCLOG file named FLHLQ.FOCLOG.DATA must be followed by at least one blank and the word *DETAIL*:

```
FLHLQ.FOCLOG.DATA DETAIL
```

Example: Requesting a Session Summary Form of the Log

This FOCUSLOG member points to the same log file as the one defined in *Requesting a Default Form of the Log* on page 261, but this request creates a session summary form of the log containing only information contained in the session segment of the FOCLOG Master File.

To create a session summary log, the FOCLOG file named FLHLQ.FOCLOG.DATA must be followed by at least one blank plus the word *SESSION*:

```
FLHLQ.FOCLOG.DATA SESSION
```

Reference: Notes on Activation and Deactivation of Logging

When FOCUS is initiated, it checks for the presence of the log file. If the FOCUSLOG file is present, and SET FOCLOG=ON is in effect, usage data is written to memory during the FOCUS session. The information stored in memory is written to the log file at FIN, during FOCUS termination, if a valid log file exists. Therefore, FOCLOG does not produce any unnecessary overhead or use additional CPU cycles for capturing usage data.

Logging is deactivated if the FOCUSLOG member is not in the concatenation of data sets allocated to DDNAME ERRORS (you can rename or remove it), or if SET FOCLOG=OFF is in effect.

Note:

- Renaming the log file itself does not deactivate logging, it only prevents the log from being copied from memory to a file. However, you can use this technique to save an old log and start a new log file. For example, if the old log file is named FLHLQ.FOCLOG.DATA, rename it to FLHLQ.FOCLOG.DATA1, and allocate a new file as FLHLQ.FOCLOG.DATA.
- Log replacement should be done as quickly as possible to avoid losing data that should be captured in the log.

Validating the FOCLOG Configuration

This step validates your FOCLOG configuration and enables you to confirm that the log file was allocated and defined properly. It does not test FOCUS functionality, which is not impacted by FOCLOG.

1. To create the validation environment, make a test copy of your FOCUS production CLIST or batch job. This CLIST should allocate the production versions of the ERRORS, MASTER, and FOCEXEC DDNAMEs.
2. Run the test CLIST or batch job to enter FOCUS.
3. Next, execute a request to populate the log.

   At the FOCUS prompt, issue the following command and press *Enter* to create a temporary FOCUS database named CAR and load it with data:

   ```
   EX FLVALPOP
   ```

   The following messages display:

   ```
   >  NEW FILE CAR      ON 09/15/2007 AT 09.06.57
    CAR      ON 09/15/2007 AT 09.06.57
    WARNING..TRANSACTIONS ARE NOT IN SAME SORT ORDER AS FOCUS FILE
    PROCESSING EFFICIENCY MAY BE DEGRADED
    TRANSACTIONS:       TOTAL =    53  ACCEPTED=    53  REJECTED=     0
    SEGMENTS:           INPUT =   102  UPDATED =     0  DELETED =     0
   ```

   You can ignore any warning messages. The important thing to note is that 53 transactions were accepted and that 102 segments were input.

   Note: If you have problems running this procedure, contact the Information Builders Customer Support Services staff.

4. Exit from FOCUS by issuing the following command and pressing *Enter*:

   ```
   FIN
   ```

   Ending the FOCUS session updates the log with the information about the procedure you executed.

5. Run the test CLIST or batch job to enter FOCUS again.
6. You will now execute a request to produce a validation report from the log.
   a. Issue the following command to allocate your log file so that you can issue a report request against it. Replace *flhlq* with the high-level qualifier for your log file data set. (**Note:** Before pressing *Enter*, make sure the data set name is the name of the log file you specified in the FOCUSLOG member of your production ERRORS.DATA library.):

      ```
      DYNAM ALLOC DD FOCLOG DA flhlq.FOCLOG.DATA SHR REU
      ```

   b. Issue the following command and then press *Enter* to report from the log:

      ```
      EX FLVALRPT
      ```

      The following messages display to indicate that the report is ready to view:

      ```
      >   NUMBER OF RECORDS IN TABLE=        3  LINES=      3
           PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
      ```

      Press *Enter* to view the report. It should be similar to the following, but display your session statistics and user ID:

```
PAGE     1

VALIDATION OF SESSION COMMANDS ISSUED

STARTDATE  SESSTART     USERID  COMMAND   FNAME    RECORDS    LINES
---------  --------     ------  -------   -----    -------    -----
2007/09/15 11:45:05.260 FLHLQ   CREATE    .              0        0
                        FLHLQ   MODIFY    CAR          102        0
                        FLHLQ   TABLE     CAR           18        1
```

      This output shows that the log is working properly. After producing the report, the procedure deallocates DDNAMEs CAR and FOCLOG.

      Note: If you have problems running this request, contact the Information Builders Customer Support Services staff.

      Press *Enter* to close the report window.

7. Exit from FOCUS by issuing the following command and then pressing *Enter*:

   ```
   FIN
   ```

You have now completed the configuration and validation process. You can now notify your FOCUS Administrator that the product is available for use.

Note: This validation procedure added data to the log file that does not reflect actual FOCUS usage at your site. Therefore, you should edit the log file to delete this data before putting the log into production. To delete the data, open the log file in the ISPF Edit Panel and delete the lines that contain the data.

Example: Sample FOCUS Session

When you enter an online FOCUS session, you see a banner similar to the following. The two carets at the bottom (> >) are the FOCUS prompt, although your site may have changed the prompt:

```
  FOCUS  7.6.3   09/15/2007  10.12.52

OBSERVED CPU:
******* CEC:  machine type N/A   model ID N/A             capacity N/A
******* LPAR: name N/A                                    capacity N/A
******* Processor AF4A Model 2066-00 Max 02  Site
LICENSED CPU(S):
******* Processor 5394 Model 9672-F0 Max 01 >Registration 57E3C86A0887
```

The following command executes the request that populates the log:

```
>  > ex flvalpop
```

The following messages display:

```
>   NEW FILE CAR      ON 09/15/2007 AT 09.06.57
 CAR      ON 09/15/2007 AT 09.06.57
 WARNING..TRANSACTIONS ARE NOT IN SAME SORT ORDER AS FOCUS FILE
 PROCESSING EFFICIENCY MAY BE DEGRADED
 TRANSACTIONS:         TOTAL =    53  ACCEPTED=    53  REJECTED=     0
 SEGMENTS:             INPUT =   102  UPDATED =     0  DELETED =     0
```

The next command ends the FOCUS session and updates the log with statistics regarding the FLVALPOP request:

```
> fin
```

Next, we run the CLIST again to go back into FOCUS:

```
  FOCUS  7.6.3   09/15/2007  10.12.52

OBSERVED CPU:
******* CEC:  machine type N/A   model ID N/A             capacity N/A
******* LPAR: name N/A                                    capacity N/A
******* Processor AF4A Model 2066-00 Max 02  Site
LICENSED CPU(S):
******* Processor 5394 Model 9672-F0 Max 01 >Registration 57E3C86A0887
```

Now we allocate the log file and issue the FLVALRPT request to view information from the log:

```
>  > dynam alloc dd foclog da flhlq.foclog.data shr reu
>  > ex flvalrpt
```

The following messages display to indicate that the report is ready to view:

```
>   NUMBER OF RECORDS IN TABLE=        3  LINES=      3

 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Pressing *Enter* displays the report:

```
PAGE     1

VALIDATION OF SESSION COMMANDS ISSUED

STARTDATE   SESSTART      USERID    COMMAND   FNAME     RECORDS    LINES
---------   --------      ------    -------   -----     -------    -----
2007/09/15  11:45:05.260  FLHLQ     CREATE    .               0        0
                          FLHLQ     MODIFY    CAR           102        0
                          FLHLQ     TABLE     CAR            18        1
```

Next, press *Enter* to close the report window and return to the FOCUS prompt.

Execute the FIN command to end the FOCUS session:

```
> fin
```

Now empty the log file (using ISPF Edit) before putting the log into production.

Running the FOCLOG Reports

FOCLOG is now configured. See *FOCLOG Reporting* on page 272, for instructions about reporting on the captured data.

Information Captured in the FOCLOG File

The FOCLOG file is a fixed format physical sequential file. FOCLOG is distributed with a Master File that describes the FOCLOG file and enables you to report from it.

The Master File has four segments:

- The session segment provides information specific to a user session.
- The command segment provides information about FOCUS commands issued during a user session.
- The master segment provides Master File names, file types, and the ID of the sink machine on which each Master File resides.

- The data set name segment provides names and locations of data sets used during a user session.

Reference: **Session Segment Information**

Field Name	Description
RECTYPE	Session segment record type (SESS) (format A4)
STARTDT	Internal date value; see DEFINE field named STARTDATE for display value (format A6)
MICROSEC	Microseconds since Jan. 1, 2000 (format A17)
SESSTART	Local time - HH:MM:SS.NN. Captured down to the nanosecond, ensuring unique session start times for users. (format A12)
USERID	User Identifier; user ID or batch job name. (format A8)
SESDURATION	Session duration, seconds. SESDURATION is displayed with 6 digits, a decimal point, then 6 digits. (format A17)
SESCPU	Session CPU time, seconds. SESCPU is displayed with 3 digits, a decimal point, then 3 digits. (format A17)
SESEXCP	Session total EXCP (format A8)
FOCREL	FOCUS release (format A16)
SITECODE	IBI sitecode (format A8)
MSO	Flag - 'Y' if MSO application (format A1)
BATCH	Flag - 'Y' if BATCH application (format A1)
CPUID	CPU INFORMATION (format A12)
OPSYS	Operating system (format A8)
OPSYSREL	OS version/release (format A8)

Field Name	Description
LPARNAM	Logical Partition Name (format A8)
IBIREG	IBI registration flag - 'Y' if IBI registration (format A1)
JOBNAME	z/OS job name (format A8)
JOBID	z/OS job ID (format A8)
MODELID	Processor Model ID (format A8)
SESZIIP	Time spent on the zIIP processor (format A17)
SESZIIPCP	Time spent on the Central Processor because the zIIP was temporarily unavailable (format A17)
SESZIIPON	Flag that indicates whether the zIIP was used during the session (format A1)
BASEDATE	DEFINE: Base date constant for date calculations (1900 DEC 31) (format YYMD)
STARTDATE	DEFINE: Start date for display (format YYMD)
STARTMONTH	DEFINE: Month of session start (format MTR)
STARTQTR	DEFINE: Quarter of session start (format YYQ)
STARTYYMTR	DEFINE: Start date with month translated (YYMTR)
STARTWEEK	DEFINE: Date of beginning of week of session start (format YYMD) **Note:** This DEFINE must be commented out when running the FOCLOG Release 1.3 version with FOCUS releases 6.0 or 6.8.
STARTHOUR	DEFINE: Hour of session start (format A2)
SERIAL_NUM	DEFINE: CPU ID serial number (format A6)
MODEL_NUM	DEFINE: CPU ID model number (format A4)

Field Name	Description
PROCESSOR_ID	DEFINE: PROCESSOR_ID combines output from the MODEL_ID and MODEL_NUM fields (A8)

Reference: Command Segment Information

Field Name	Description
RECTYPE	Command segment record type (CMDS) (format A4)
COMMAND	FOCUS command (for example, TABLE, MODIFY) (format A8)
FOCEXEC	FOCEXEC name if run by an EXEC command (format A8)
LINENUM	Line number of command in the FOCEXEC (format A6)
CMDSTART	Seconds since start of session. CMDSTART is displayed with 5 digits, a decimal point, then 3 digits. (format A17)
CMDCPU	CPU time to execute the command. CMDCPU is displayed with 5 digits, a decimal point, then 3 digits (format A17)
CMDDURATION	Duration of command, in seconds. CMDDURATION is displayed with 5 digits, a decimal point, then 3 digits (format A17)
BASEIO	Statistics of last command issued: Baseio (format A7)
SORTIO	Statistics of last command issued: Sortio (format A7)
RECORDS	Statistics of last command issued: Records for TABLE, input for MODIFY (format A7)
LINES	Statistics of last command issued: Lines for TABLE, changed for MODIFY (format A7)
READS	Statistics of last command issued: Reads for TABLE, deleted for MODIFY (format A7)

Field Name	Description
CMDEXCP	Statistics of last command issued: EXCPs for command (format A7)
HLRECL	Hold file lrecl (format A5)
POOLED	Flag - 'Y' if pooled tables used (format A1)
EXTSORT	Flag - 'Y' if external sort used (format A1)
OUTFLAG	Output format numeric value (internal value; see DEFINE field OUTPUT_TYPE for display value) (format A3)
OFFLINE	Flag - 'Y' if OFFLINE output (format A1)
FOCEXECDSN	FOCEXEC file library name - populated in long log format only. (format A44)
CMDZIIP	Time a FOCUS command spent on the zIIP processor (format A17)
CMDZIIPCP	Time a FOCUS command spent on the Central Processor because the zIIP was temporarily unavailable (format A17)
CMDZIIPON	A flag that indicates whether the zIIP processor was used during execution of the command (format A1)
OUTPUT_TYPE	DEFINE: output type name (format A8). Decode OUTFLAG table
OUTVOLUME	DEFINE: calculated file size in bytes (format D15). IF OUTPUT_TYPE is HOLD, SAVE, SAVB, or PCHOLD THEN HLRECL x LINES ELSE 0.
FOCEXEC	DEFINE (format A8): IF FOCEXEC LT 'A' THEN ' ', ELSE FOCEXEC

Reference: Master Segment Information

Field Name	Description
RECTYPE	Master segment record type (MASS) (format A4)
FNAME	Master File name (format A8)
SUFF	File suffix (such as FOC or FIX) (format A8)
SINKID	Sink identifier (if SU used, otherwise blank) (format A8)
MASTERDSN	Master File library name - populated in long log format only. (format A44)
ACCESSDSN	Access File library name - populated in long log format only. (format A44). For information about capturing ACCESSDSN, see *Capturing the Access File Data Set Name* on page 270.
FILE_TYPE	DEFINE: Translates SUFF into a more understandable name (format A8)

Reference: Capturing the Access File Data Set Name

If you are using an Access File DDNAME other than ACCESS, you may be able to allocate those data sets to DDNAME ACCESS instead, in order to capture the Access File data set names in the log file.

Reference: Data Set Segment Information

Field Name	Description
RECTYPE	Data set name record type (DSNS) (format A4)
DDNAME	z/OS DDNAME (format A8)
DSNAME	Data set name (format A44)
DDEXCP	EXCPs for ddname (format A7)

Field Name	Description
DSNORD	Order number of concatenation (format A3)
DEVFLAG	Flag - 'Y' if possible missing EXCPs (format A1)
DSNTEMP	Flag - 'Y' if dsn is a temp file (format A1)

Reference: **Master File Structure Diagram**

The following diagram shows the structure of the FOCLOG Master File and the relationships between its segments. It lists the first four fields in each segment.

```
NUMBER OF ERRORS=     0
NUMBER OF SEGMENTS=   4  ( REAL=    4  VIRTUAL=   0 )
NUMBER OF FIELDS=    55  INDEXES=   0  FILES=     1
NUMBER OF DEFINES=    7
TOTAL LENGTH OF ALL FIELDS=  509
SECTION 01
            STRUCTURE OF FIX      FILE FOCLOG   ON 05/07/04 AT 13.23.43

         SESS
 01      S0
**************
*RECTYPE     **
*STARTDT     **
*MICROSEC    **
*SESSTART    **
*            **
***************
 ***************
       I
       +-----------------+
       I                 I
       I CMD             I DSN
 02    I N          04   I N
**************    **************
*RECTYPE     **    *RECTYPE     **
*COMMAND     **    *DDNAME      **
*FOCEXEC     **    *DSNAME      **
*LINENUM     **    *DDEXCP      **
*            **    *            **
***************   ***************
 ***************    **************
       I
       I
       I
       I MAS
 03    I N
**************
*RECTYPE     **
*FNAME       **
*SUFF        **
*SINKID      **
*            **
***************
 **************
```

FOCLOG Reporting

FOCLOG comes packaged with a Master File that gives FOCUS access to the log file. It also comes with a menu driven interface with usage reports that you can start running as soon as you decide you have collected enough usage data.

Using the Menu-Driven FOCLOG Reporting Interface

The menu-driven FOCLOG reporting interface provides a review of usage patterns and detailed analysis of the FOCLOG file. Overlap between the summary and detailed report contents support easy analysis of the usage data. Run-time options enable easy adjustment of the collection and aggregation periods and sort criteria to aid in analyzing apparent anomalies.

The reports are organized into dimensions that group similar types of reports and concepts of analysis.

Reference: Specifying Values Used in Report Generation

Before running the reporting interface, you must edit the FLPROF FOCEXEC file to specify the company name you want to display in the report headings and to allocate your FOCLOG log file.

The following is a listing of the FLPROF FOCEXEC:

```
-*************************************************************************
-*
-* INFORMATION BUILDERS INC.
-* FOCLOG STATISTICAL REPORTS
-*
-* PROFILE FOR FOCLOG REPORTING
-*                                                          @MFSM_NOPROLOG@
-*************************************************************************


-* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
-* PUT YOUR COMPANY NAME HERE AS IT SHOULD APPEAR ON ALL REPORTS.
-SET &&COMPANY  = 'INFORMATION BUILDERS INC.';
-*
-IF &FOCMODE EQ 'TSO' OR 'MVS' GOTO MVS_ALLOC ;
-*                    VM/CMS USERS
-* CHANGE THE FOLLOWING LINE TO REFLECT THE SFS DIRECTORY NAME
-*        YOU CHOSE FOR YOUR FOCLOG FILE.
CMS FILEDEF FOCLOG DISK VMSYSU:FOCLOG.DATA
-GOTO CONT
-MVS_ALLOC
-*                   MVS/TSO USERS
-* CHANGE THE FOLLOWING LINE TO REFLECT THE DATASET NAME
-*        YOU CHOSE FOR YOUR FOCLOG FILE.
DYNAM ALLOC FILE FOCLOG DA FLHLQ.FOCLOG.DATA SHR REUSE
-RUN
-CONT
-* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Make the following changes:

- Replace 'INFORMATION BUILDERS, INC.' with your company name in the following line:

  ```
  -SET &&COMPANY  = 'INFORMATION BUILDERS INC.';
  ```

- Replace FLHLQ.FOCLOG.DATA with the fully qualified name of your FOCLOG log file in the following line:

```
DYNAM ALLOC FILE FOCLOG DA FLHLQ.FOCLOG.DATA SHR REUSE
```

Save the edited version of FLPROF FOCEXEC before running the reports.

Procedure: How to Invoke the FOCLOG Reporting Interface

From FOCUS, issue the following command to display the report selection menu:

```
EX FLMENU
```

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

DIMENSION:  USER    FILE    PROCEDURE    USAGE    CUSTOM    ADHOC
            ****

     101 - DURATION OF ONLINE SESSIONS
     102 - HIGHEST CPU-CONSUMING USERS AND JOBS
     103 - MOST ONLINE SESSIONS
     104 - TOTAL MSO SESSIONS
     105 - TOTAL FOCUS USERS
     106 - ATTEMPT TO GROUP USERS BY 4-CHAR PREFIX
     107 - SITE-WIDE FOCUS USAGE TREND
     108 - BATCH JOB DETAILS
     109 - ONLINE SESSION DETAILS


     SELECT:

F3=EXIT    F7=PREVIOUS SCREEN    F8=MORE REPORTS    F12=HELP
```

Use the PF8 and PF7 keys to move forward and backward through the report selection menus for the following five reporting dimensions listed at the top of the screen:

- **USER.** These reports describe user-oriented activity on the system.
- **FILE.** These reports describe files accessed by FOCUS.
- **PROCEDURE.** These reports describe FOCUS programs running on the system.
- **USAGE.** These reports describe FOCUS activity.
- **CUSTOM.** These reports are provided by the user. For information on creating your own reports and adding them to the reporting interface, see *How to Build and Catalog a Custom FOCLOG Report* on page 278.

To select a specific report, enter its three-digit report number. If you know a report's three-digit number, you can enter that number from any screen.

Tip: If you are on a screen and want to run a report listed on that screen, you can enter the one or two-digit number to the right of the leftmost digit (which is the screen number). For example, if you are on screen 1, you can run report 109 by entering 9 or 09.

If you want to enter FOCUS from the interface in order to run your own requests against the FOCLOG file, select the ADHOC dimension.

Use the function keys as described at the bottom of each screen. Look for helpful messages that appear below the function key line.

Once you have selected a report, an options screen displays:

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

**** REPORT SELECTED: 101  ****

SELECT DATE RANGE OF REPORT (YYYY/MM/DD):     2009/12/01  TO 2010/03/22

AGGREGATE BY (Q)UARTER, (M)ONTH OR (W)EEK:    M

SEND REPORT TO (S)CREEN, (P)RINTER OR (H)OLD? S
               IF (H)OLD, SELECT HOLD FORMAT:          AS

CHOOSE OPTIONAL SORT FROM (NONE AVAILABLE)  : N

RUN LIMITED DATA TO VIEW REPORT LAYOUT (Y/N): N

ENTER=RUN REPORT   F3=BACK TO REPORT SELECTION   F12=HELP
```

You can specify the following options on this screen:

- The date range covered. Supply the dates in YYYY/MM/DD format, using the default values as a guide. The default is a three-month range from the current date backwards.
- The aggregation period (Quarter, Month, Week). This applies to reports that have a calendar sort level and can accommodate the selection.
- The report destination (Screen, Print, HOLD file). Enter:

  S to see the report on your screen.

  P to send the report to your printer.

  H to store the report results in a HOLD file. By default, the format type will be a BINARY HOLD file. You can specify a different format and an AS phrase to supply a name other than the default name HOLD.

- Sort options, if any are available for that report. The first sort is generally the time period. Some reports can be additionally sorted on other columns, such as duration, CPU, EXCP, records, and lines. Only the sorts applicable to that report are shown. If none are available, NONE AVAILABLE displays. Use the first letter of the word to select that sort column, or if you do not want an additional sort level, either leave it blank or enter N (None).
- The option of making a limited request to review report layout. Enter Y at this prompt to see a version of the report with limited data. A report may not display at all if no data matches the criteria for selecting the limited records.

The date period you select for the first request in a session remains in effect until you change it, which you can do on any report selection screen. This is also true for aggregation periods and report destinations.

Example: Running a FOCLOG Report

The following example runs FOCLOG report FLRPT301.

First, make sure you have access to the FOCLOG log file and that you added the appropriate DYNAM command in the FLPROF FOCEXEC file.

Then, issue the following command to execute the reporting interface:

```
ex flmenu
```

The following screen displays:

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

DIMENSION:  USER    FILE    PROCEDURE    USAGE    CUSTOM    ADHOC
            ****

     101 - DURATION OF ONLINE SESSIONS
     102 - HIGHEST CPU-CONSUMING USERS AND JOBS
     103 - MOST ONLINE SESSIONS
     104 - TOTAL MSO SESSIONS
     105 - TOTAL FOCUS USERS
     106 - ATTEMPT TO GROUP USERS BY 4-CHAR PREFIX
     107 - SITE-WIDE FOCUS USAGE TREND
     108 - BATCH JOB DETAILS
     109 - ONLINE SESSION DETAILS


     SELECT:

F3=EXIT   F7=PREVIOUS SCREEN   F8=MORE REPORTS   F12=HELP
```

Press *F8* twice to open the PROCEDURE screen:

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

 DIMENSION:  USER    FILE    PROCEDURE    USAGE    CUSTOM    ADHOC
                             *********

      301 - MOST FREQUENTY RUN PROCEDURES
      302 - FOCUS COMMAND USAGE









      SELECT:

 F3=EXIT    F7=PREVIOUS SCREEN    F8=MORE REPORTS    F12=HELP
```

Enter the number 301 in the SELECT field to run FLRPT301. The Options screen opens:

```
                  MAINFRAME FOCUS UTILIZATION ANALYSIS

**** REPORT SELECTED: 301  ****

SELECT DATE RANGE OF REPORT (YYYY/MM/DD):     2010/02/01  TO 2010/05/03

AGGREGATE BY (Q)UARTER, (M)ONTH OR (W)EEK:    M

SEND REPORT TO (S)CREEN, (P)RINTER OR (H)OLD? S
               IF (H)OLD, SELECT HOLD FORMAT:           AS

CHOOSE OPTIONAL SORT FROM (NONE AVAILABLE)  : N

RUN LIMITED DATA TO VIEW REPORT LAYOUT (Y/N): N

ENTER=RUN REPORT   F3=BACK TO REPORT SELECTION   F12=HELP
```

Accept the default options and run the report by pressing *Enter*. A report similar to the following displays:

```
                MAINFRAME FOCUS UTILIZATION ANALYSIS
                      INFORMATION BUILDERS INC.
                FEBRUARY  1, 2010 -  MAY  3, 2010
                    MOST FREQUENTLY RUN PROCEDURES

                                          CPU USAGE  ZIIP ON CP  ZIIP USAGE
     MONTH      PROCEDURE FREQUENCY        HH:MM:SS    HH:MM:SS    HH:MM:SS
--------------- --------- --------- ----------- ----------- -----------
2010, FEBRUARY  FORMATS          15       00:00       00:00       00:00
                MORE             15       00:00       00:00       00:00
                VSAM              4       00:02       00:00       00:00
                VALIDATE          6       00:00       00:00       00:00
                MATCHFIX          4       00:00       00:00       00:00
                USEBANK          17       00:02       00:00       00:00
   32 PROCEDURES EXECUTED     74 TIMES (TOP 30 SHOWN)

--------------------------------------------------------------------------
2010, MARCH     COMBINE           5       00:00       00:00       00:00
                AMEX3             4       00:03       00:00       00:22
                LOOKUP            4       00:01       00:00       00:10


        PAGE    1               2010/05/03 16.04.28    FLRPT301
                                                                   MORE
```

Press *Enter* to scroll through subsequent pages of the report. At the last page of the report, press *Enter* to return to the menu screen.

Procedure: How to Build and Catalog a Custom FOCLOG Report

1. Write and debug a FOCEXEC that reports against the FOCLOG file. The author is responsible for code and results.
2. Store the FOCEXEC with a name of the following form:

   FLRPT5*nn*

   where:

   nn

   Is the next available two-digit number on the CUSTOM screen. For example, the ninth custom report should be named FLRPT509. The screen supports up to 10 reports.
3. Have your FOCLOG Administrator edit the FLMENU FOCEXEC to:
   a. Adjust the CRTFORM of the CUSTOM screen with the name of your report at the proper number.
   b. Then, after the following line, change the associated 9*xx* to 5*xx*:

      ```
      '-* REPORT NUMBER VALIDATION'
      ```

If you need any SET commands in your FOCEXEC, reset them at the end of your routine. To do that, first capture the current setting with the following command:

```
? SET setname &HOLDIT
```

Then restore it at the end of the routine with:

```
SET setname=&HOLDIT
```

Report Layouts

The packaged reports all have standard four-line headings:

```
MAINFRAME FOCUS UTILIZATION ANALYSIS
COMPANY NAME
DATES COVERED (FROM - TO)
FOCLOG REPORT NAME (generated by your selection)
```

Report footings contain the page number, the date and time the report was requested and the internal name of the report corresponding to its number on the selection screen.

Report Contents

Each report examines an aspect of FOCUS usage at your site. There is considerable overlap among reports, as some present high-level usage summaries while others provide underlying details, potentially generating tens or even hundreds of pages.

Use the detail level reports to examine suspicious or irregular value clues on the summary reports.

Sort Options

Columns are sorted consistently across reports. Date periods are sorted in ascending chronological order, names are sorted alphabetically, and numeric columns are typically sorted high to low where the numbers of most interest tend to be the higher ones.

When optional sorts are invoked, the notation "* * SORT* *" appears at the top of the sort column indicating its selection by the user as opposed to a default.

Report Descriptions

Each of the following tables describes the reports available for a specific dimension. Each report has an internal name of the form:

```
FLRPTnnn
```

where:

nnn

Is the three-digit report number on the menu screen.

Reference: **User Reports - 100 Series**

The 100 Series reports show user activities and the resources consumed, permitting examination of usage by individuals, groups, usage types (online vs. batch), or a site-wide trend analysis.

No.	Report Title and Contents	Interpreting Report Output
101	**Duration of Online Sessions** summarizes the length of time users were logged on in predefined ranges of hours (<1, 1-4, 4-8, 8-12, >12). Only ranges with contents are displayed. ❑ The <1 hour group is usually ignored. ❑ 1-4 hours is effectively a morning or afternoon session, realizing that many users log off at lunch time, which creates two 1-4 hour sessions. ❑ 4-8 hours is a normal work day. ❑ 8-12 hours may show a dedicated worker. ❑ >12 hours may be a terminal left on overnight.	Look for the >12 hours category, which often indicates IDs left on overnight, exorbitant usage, or a process running on an unattended ID. Many other reports show the breakdown of Duration to further investigate these situations. For example, sort report 109 by Duration to see who was logged on for the various periods.
102	**Highest CPU-Consuming Users and Jobs** identifies users running procedures that absorbed the most CPU and zIIP usage during the period. The report is sorted by CPU usage for Batch jobs and Online sessions.	High CPU and zIIP usage (particularly for online sessions) may indicate abuse or, more likely, users whose needs should be evaluated for possible efficiency improvement.

No.	Report Title and Contents	Interpreting Report Output
103	**Most Online Sessions** summarizes usage trends over the period by number of sessions (grouped by tens) and most frequent users in each.	Spikes identify heavy usage periods, such as end-of-month, quarter, or year, or special promotions or events. Upward trends by period may indicate other symptoms. Run Report 108 or 109 to see details of a specific user's activity in the higher categories.
104	**Total MSO Sessions** shows, by period, what part of FOCUS online usage is performed using MSO (for sites running MSO).	A significant percentage generally indicates a company's efficient use of shared resources provided by MSO.
105	**Total FOCUS Users** rolls up overall FOCUS usage across the site for each period. The Active Users and Total Jobs columns imply average usage per user, and the CPU and zIIP columns track the actual FOCUS processing performed during the period.	Look for spikes (positive or negative) and progressive trends in the numbers. Many other reports break down these summarized numbers.
106	**Attempt to Group Users by 4-Char Prefix** assumes that the first four characters of the user ID roughly corresponds to definable groups in an organization. If that assumption is true for your company, this report may be of value. The report shows the number of users with that prefix and the CPU and zIIP used by that group. Re-sorting by CPU could point to the largest group of FOCUS users on the site.	Assuming the first four user ID characters are significant workgroup differentiators, this report can be useful for comparing relative usage loads. Alternatively, see *Information Captured in the FOCLOG File* on page 265 for a description of the FOCLOG Master File in order to develop your own schemes for collapsing user IDs into local groups.

No.	Report Title and Contents	Interpreting Report Output
107	**Site-Wide FOCUS Usage Trend** summarizes site-wide FOCUS usage for each period, broken out by batch and online usage and by Reports (extracts) versus Updates (data changes).	Look for significant usage spikes in CPU and zIIP or number of runs, and possible imbalances between reports and update operations.
108	**Batch Job Details** describes every batch job during the period, showing the timestamp, who ran it, the LPAR FOCUS used, the job duration, and CPU and zIIP utilized.	This report could be hundreds or thousands of pages long. Sort by Duration or CPU to review largest jobs first.
109	**Online Session Details** describes every online session run during the period, showing its start time, who logged on, the LPAR FOCUS used, job duration, CPU, and zIIP utilized.	This report could be hundreds or thousands of pages long. Sort by Duration or CPU to see the most CPU-intensive sessions first. Similar to FLRPT108 for batch usage.

Reference: File Dimension Reports - 200 Series

The 200 Series reports identify key files and show when and how they were used.

No.	Report Title and Contents	Interpreting Report Output
201	**Possible Extract Files Being Used** identifies the largest flat files being used. Flat files are non-keyed sequential files rather than structured databases. They are typically created by other applications or by extracts from local databases. Extract files are often CPU savers in that they require fewer resources than repeatedly retrieving data from the main file. Not all files on this report are 'extract' files; the report only knows that they are SUFFIX=FIX files, so some knowledge of the origin and use of the files are critical to the evaluation.	Low use of extract files should be investigated for potential removal or merging. Even high usage could point to the need to evaluate a better way to supply that data. Extract files often unknowingly grow excessively large over time, so a point of re-evaluation may be warranted.
202	**Files Originating Large HOLD File Extracts** shows databases from which large extracts are generated. Frequent creation may point to the need to evaluate the creation strategy.	Confirm acceptability of large HOLD files.
203	**Files Used by Users** shows the files accessed during the period by each user. Sort the report by Records, Lines, or CPU to see the most used files.	Use to investigate a user ID whose activity was highlighted on another report.
204	**Batch and Online File Usage** shows relative FOCUS online and batch usage by file type. This report concentrates on the type of file rather than the file name itself.	Usage at both ends of the spectrum may be of interest, depending on site conditions. Re-sort by Duration, CPU, or EXCP to review impact by category.

No.	Report Title and Contents	Interpreting Report Output
205	**Reports and Transactions Against Data Sources** details usage of specific data sources during each period. Sorted by period and file name.	Note the effect of file types on statistical columns. Scroll right to view the Batch Updates column, which, along with the Batch Reports column, indicates the number of times the file was accessed during the period. Re-sort by Duration, CPU, or EXCP to review most used files by those categories.
206	**Files Using the Most CPU on Reports** rolls up file usage across the entire period. Sorted by file name, you may re-sort by CPU, Records, or Lines to observe the most used files. Use FLRPT205 to break down the file usage by smaller periods.	The Rollup shows the degree to which the data extracted was aggregated on the report versus printed row by row as raw data. This implies output volume and how that file is being used. The higher the Rollup, the better.
207	**Data Source Types Accessed by FOCUS** summarizes FOCUS use by file type.	Run Report FLRPT206 to assess usage by individual file.
208	**Most Records Extracted During Online Sessions** summarizes users who extracted very large volumes of data on a regular basis, grouping record extracts in 100,000 record increments. It shows number of occurrences per group.	High numbers of occurrences might warrant a review of usage or indicate the need for a better way to access the data regularly.

Reference: Procedure Dimension Reports - 300 Series

No.	Report Title and Contents	Interpreting Report Output
301	**Most Frequently Run Procedures** lists the 30 most frequently executed procedures and their corresponding CPU and zIIP usage sorted by frequency. You can re-sort by CPU to see the largest running procedures. Run FLRPT303 to list all procedures run during the period and review full usage details.	Frequently run procedures can generally be construed to be clearly valuable to the company. As such, you may want to get the most out of your investment by examining such procedures for efficiency, particularly if the CPU or CPU plus zIIP is high.
302	**FOCUS Command Usage** presents a broad summary of FOCUS usage across your site. Typically, TABLE, that is, data extracts into reports or extract files will be the most used command. MODIFY indicates loads of, or transactions against, data files. Other major FOCUS functions are shown if they are used.	Re-sort by Duration, CPU, or EXCP to identify the most used commands by category.
303	**Procedure Run Details** lists every procedure run during the period showing the frequency, duration, CPU, zIIP, and EXCP demands (and relative percent of each) within the period.	Re-sort by Duration, CPU, or EXCP to identify the most resource-intensive or widely-used procedures. Run FLRPT301 for a summary of the top 30 procedures.

No.	Report Title and Contents	Interpreting Report Output
304	**Candidate Procedures for Pooled Tables Adaptation** uncovers file usage patterns that appear to lend themselves to pooling, which could provide substantial CPU and zIIP savings. This report has the potential of identifying procedures that can experience large CPU and zIIP savings for relatively little investment.	Pooled Tables performs a single read of a data source for any number of consecutive reports against that source, delivering huge savings in I/O, and in CPU, zIIP, and elapsed times. Information Builders can help you evaluate the viability of pooling procedures listed on this report. Contact your local Information Builders representative for details about Pooled Tables.

Reference: Usage Dimension Reports - 400 Series

No.	Report name and contents	Interpreting Report Output
401	**All LPARS Running FOCUS** summarizes FOCUS usage/per period in each LPAR where FOCUS activity was recorded.	This is the only report of FOCUS activity by LPAR. You can re-sort the results by Duration or CPU.

No.	Report name and contents	Interpreting Report Output
402	**Output Formats Used by Reports** summarizes report destinations used during the period, revealing numbers of records extracted and lines output and the degree of aggregation (compression of application output). Many reports go directly to the user's screen or are sent to FOCUS HOLD or SAVE files (the General category), and some are sent to other output formats available from FOCUS, listed below that. Only those formats used are listed. Re-sort by Records, Lines, or CPU to see the most common or intensive destination for FOCUS output.	High aggregation ratios show data is being aggregated into more readily interpretable information. A low ratio for SCREEN implies that the raw data is displayed potentially hundreds of screens deep, which is invariably impenetrable and should be re-evaluated for usability. Non-FOCUS output formats imply FOCUS is supplying data for third-party databases and analytical tools, such as Excel or PDF.
403	**Daily User Activity Detail** shows daily activity for every online session for every user during the period. **Caution:** This report could easily run hundreds or thousands of pages.	This report provides the detail behind FLRPT102, Highest CPU and zIIP-Consuming Users and Jobs. Look for repeated excessive expenses, which may in fact be perfectly valid high product usage. Re-sort by Records, Lines, Duration, or CPU to look for highest usage or minimal usage.
404	**Trend of Daily FOCUS Sessions (Report and Graph)** bar graph summarizes the number of batch sessions per day.	Review usage trends and analyze spikes. Days without sessions, including weekends, are not shown.

No.	Report name and contents	Interpreting Report Output
405	**Hourly CPU Usage Accumulated (Graph)** bar graph accumulates CPU and zIIP usage for each hour of the day, rolling together all days in the period. That is, a bar represents all of the usage accumulated on all days in the period for that particular hour. Hours are sorted from midnight to midnight; hours with no activity do not display. An hour shown represents the minutes in that hour, so '2am' represents 2am-3am.	Confirm the industry-typical trend of highest usage during workday hours and during peak times of overnight batch runs. Other spikes may be shift-related. Restrict the date range to one day to examine CPU and zIIP usage on a particular day.
406	**Possible Large Paper-Output Reports (100+ pages)** details procedures generating large offline print files. The report sorts the list by the largest reports in terms of approximate number of pages generated (assuming 40 lines of output per page, allowing for some typical lines of heading, footing, column heading, subfoots, blank lines, and so on). The report shows the procedure that ran the report and the (main) file against which the extract was done, as well as the number of times the report was so generated.	Confirm acceptable usage and number of runs. The largest printed reports are the ones to investigate.
407	**Daily Batch/Online CPU Utilization by Shift** categorizes CPU and zIIP utilization by operating mode (online versus batch sessions) by approximate shifts (from 8AM to 6PM and from 6PM to 8AM) for each day in the report period.	A useful report for confirming performance is within normal bounds and if your CPU and zIIP usage is inline with industry trends.

No.	Report name and contents	Interpreting Report Output
408	**Long-Running Sessions** is an elapsed-time summary that rolls up all batch jobs and online sessions during the period, displaying the longest and average durations and number of runs within two ranges of hours (>2 and >7).	Average numbers from this report should be useful in evaluating the data on other reports.

Chapter 9

Configuring FOCUS for National Language Support Services

This chapter describes how to add code pages to the FOCUS configuration and change the language settings in effect when FOCUS is invoked.

In this chapter:

- Introduction to FOCUS National Language Support (NLS) Services
- Detailed NLS Configuration Steps
- Advanced NLS Configuration Options
- Using the NLS Configuration Files
- Using the TSGU to Generate New Transcoding Tables

Introduction to FOCUS National Language Support (NLS) Services

NLS services enable FOCUS to support many different languages. Each supported language has an associated code page. In FOCUS, the mapping of graphic characters from one code page into the graphic characters of another is managed by the FOCUS NLS translation component.

FOCUS comes with the Transcoding Services Generation Utility (TSGU) program that you execute to generate the NLS transcoding table file based on the code page list of your site.

There are no additional allocations you need to issue in order to add or edit your code page and language settings. You can run the TSGU from FOCUS using the FOCUS CLIST or JCL created during installation.

Note: NLS is only supported if your terminal emulator is IBM Personal Communications (PC3270).

You can customize your NLS FOCUS configuration to:

- **Change your code page settings.** You can specify which code page transcoding tables you want to configure for FOCUS. For more information, see *Adding New or Alternate Code Pages for FOCUS* on page 296.
- **Customize NLS default characteristics.** You can change the FOCUS code page and the language FOCUS uses for error messages. For more information, see *Configuring FOCUS To Use a New Code Page* on page 299.

- **Customize monocasing.** You can customize the conversion of letters from lowercase to uppercase or uppercase to lowercase. For more information, see *Configuring Customized FOCUS NLS Monocasing* on page 300.
- **Customize sorting.** You can customize the sort sequence used by FOCUS. For more information, see *Configuring Customized FOCUS NLS Sort Sequences* on page 302.

Note:

- Customization of your NLS configuration is only necessary if you require support for alternate or additional code pages, or custom monocasing and sorting.
- If you modified your code page configuration in a FOCUS release prior to FOCUS 7.7, and you want to keep the same language and code page configuration, you can copy the CPCODEPG file, any CP*nnnnn* files that were modified, and the NLSCFG file from the FOCCTL.DATA library in the prior FOCUS release to the FOCUS 7.7 library named *install_hlq*.CONF.CFG.
- When you upgrade from a FOCUS 7.7 release to a new release of FOCUS, and you want to keep the same code page and language configuration, you can copy your code page configuration files from the *install_hlq*.CONF.CFG library in the old release of FOCUS to the *install_hlq*.CONF.CFG library in the new release of FOCUS.
- If you want to customize your NLS configuration and the member NLSCFG does not exist in the concatenation of data sets allocated to DDNAME ERRORS, create a member named NLSCFG in the *install_hlq*.CONF.CFG library.
- If you want to configure new code page and language settings for the new release of FOCUS, follow the instructions in *Adding New or Alternate Code Pages for FOCUS* on page 296.

Example: Sample NLS Configuration Session

This example adds French code page 297 to the FOCUS configuration. The instructions in the remainder of the chapter describe the process in detail but, in most cases, it is a simple process.

1. Copy the following line for French code page 297 into the CPCODEPG file.

   ```
   CP00297   E SBCS  IBM MF France
   ```

2. Enter FOCUS and issue the following command to generate the transcoding tables.

   ```
   TSGU
   ```

3. Create the NLSCFG file with the following settings to change the default code page and language settings.

```
LANG=FRE
CODE_PAGE = 297
DATEOUTPUT=DEFAULT
COLLATION=CODEPAGE
```

The detailed instructions in *Detailed NLS Configuration Steps* on page 296 first have you copy all files that may be modified or used for copying to a new location in order to leave the original files intact.

Overview of Steps for Configuring Code Page Settings

You must perform the following steps to change your code page configuration:

1. Review your current code page list and decide whether you need alternate or additional code pages.
2. Update the code page generation list of your site with the code page information required for those alternate or additional code pages.
3. If you need to change any of the code pages to conform to the requirements of your site or for the purpose of monocasing or sorting, edit the appropriate code page definition files.
4. Start FOCUS and run the TSGU to generate the new transcoding tables based on the updated code page generation list.
5. If you generated a new FOCUS code page in step 4, configure FOCUS to use the new code page and/or language.
6. While logged out of z/OS, change the session parameters in your PC3270 configuration to use the same code page as the code page to be used by FOCUS.
7. Verify the configuration by logging back in to z/OS and invoking FOCUS.
   a. Issue the ? LANG query command to see the language and code page settings in effect. For example:

```
        NATIONAL LANGUAGE INFORMATION

Language                033/FRENCH      (FRE,fr)
Code Page               297
Client Code Page        297
Dollar                  5B($)
Lowcase alphabet        YES
Decimal notation        OFF(.)
Currency symbol         $
Date/Time format        EDA
NLS sort                NO(BINARY)
NLS upcase/lowcase      NO
NLS Control Characters  OFF
DBCS Flag               OFF(SBCS)
```

   b. Display data that has characters from the new code page to see that they display properly.

During NLS configuration, some of the distributed NLS files need to be edited. In order to leave the original files intact, copy the code pages that you will edit to *install_hlq*.CONF.CFG and edit those copies.

Note: If you reinstall FOCUS over your current production FOCUS libraries, using the same names, the configuration changes will be overwritten. In this case, you should copy these files to another location before reinstalling, and then copy them back after the new installation process is complete.

Overview of NLS Configuration Files

The following table describes the NLS configuration files distributed with FOCUS 7.7 in alias library *install_hlq*.ERRNLS.DATA (physical library *install_hlq*.HOME.ERR):

Member	Description and Use
CPCODEPG	**Description:** Code page generation list. This file is a list of currently active code pages. **Use:** You must update this file in order to change your NLS configuration.
CPCPALL	**Description:** List of language codes and associated code pages. **Use:** This file is a reference file containing a list of language codes and code pages for all languages supported for use with FOCUS.
CP*nnnnn*	**Description:** Code page definition file. This file contains information on each code point value in the code page. There is a code page definition file for every code page listed in the known code page file (CPXCPTBL). **Use:** This file will not be updated unless you want to change any character within the code page character definitions.
CPXCPTBL	**Description:** Known code page list. This file contains a list of enabled code pages for TIBCO FOCUS products. **Use:** This file is a reference file for finding alternate or additional code pages to add to CPCODEPG.

The following table describes the NLS configuration files that may be created by the TSGU (the utility that generates the tables required for your NLS configuration). These files are generated in the physical library *install_hlq*.CONF.CFG:

Member	Description
CASETBL	**Monocasing table.** This file contains the monocasing tables, which are generated using the TSGU and are based on the code page generation list (CPCODEPG). You only need to generate this table if you customized the monocasing options in any of the code page definition files.
SORTTBL	**Sorting table.** This file contains the sorting tables, which are generated using the TSGU and are based on the contents of the code page generation list (CPCODEPG). You only need to generate this table if you customized the sorting options in any of the code page definition files.
TRANTBL	**Transcoding table.** This file contains the transcoding tables, which are generated using the TSGU and are based on the contents of the code page generation list (CPCODEPG).

You also need to create an NLS configuration file, member NLSCFG in library *install_hlq*.CONF.CFG, if you want to start FOCUS using a non-default language or code page.

You must also configure your terminal emulator software to use the same code page as FOCUS.

The FOCUS Default Code Page Configuration

Code page settings are reflected in the code page generation list file *install_hlq*.ERRNLS.DATA(CPCODEPG). This file contains code page settings for FOCUS and it is used by the TSGU to generate the transcoding tables. To identify your current code page settings, view the code page generation list file (CPCODEPG).

During the FOCUS installation, FOCUS is set up by default with the US EBCDIC code page 37. The code page configuration for FOCUS is set up to support the following default code pages:

- CP00037 E SBCS US IBM MF EBCDIC code
- CP00437 A SBCS US PC ASCII code

- CP00137 A SBCS ANSI Character Set for MS-Windows
- CP01047 E SBCS IBM MF Open Systems (Latin 1)
- CP65001 A UTF8 Unicode (UTF-8)

You can add additional code pages to this configuration, replace existing code pages with alternates, or change the properties in a code page file to conform to the standards of your site or to modify the monocasing or sorting properties of the code page.

Detailed NLS Configuration Steps

This section describes how to add code pages to the FOCUS configuration, implement default language settings, configure PC3270 for the same code page as FOCUS, and verify the configuration.

Adding New or Alternate Code Pages for FOCUS

Review your code page configuration requirements. If you did not previously modify your code page configuration but need to do so in FOCUS 7.7, follow the instructions in this section.

If you require customized monocasing or sorting for one or more of your configured code pages, edit the code page definition file for those code pages as described in *Advanced NLS Configuration Options* on page 300.

Procedure: How to Add New or Alternate Code Pages to FOCUS

1. Copy member CPCODEPG (list of code pages to be configured), any CP*nnnnn* files that you may want to modify, member CPCPALL (list of languages, language codes, and their associated code pages), and member CPXCPTBL (known code page list) files from *install_hlq*.ERRNLS.DATA to *install_hlq*.CONF.CFG.

   All changes will be made in the *install_hlq*.CONF.CFG library so that the original files remain intact.

   Note: If you reinstall FOCUS over your current production FOCUS libraries, using the same names, the configuration changes will be overwritten. In this case, you should copy these files to another location before reinstalling, and then copy them back after the new installation process is complete.

2. Browse member CPCPALL in the *install_hlq*.CONF.CFG library to determine which code pages are available and supported for the language you want to configure.

The following is a portion of the CPCPALL file that contains the language abbreviation and supported code pages for French:

```
* LANG=FRE
cp297
cp863
cp137
cp1047
cp37
cp65001
BR
```

3. Copy additional or alternate code pages into CPCODEPG from the known code page file, CPXCPTBL.

   Add the information for each additional or alternate code page on a separate line. Note that only the characters CP and the code page number (for example, CP00297) are required to generate the new transcoding tables. The maximum number of code page entries in the file is 16.

   a. Open the known code page file, member CPXCPTBL in the library *install_hlq*.CONF.CFG.

      The following is a portion of the CPXCPTBL file that contains the code page information for French code page 297 (CP00297):

      ```
      CP00281   E SBCS  IBM MF Japanese English
      CP00284   E SBCS  IBM MF Spain/Latin America
      CP00285   E SBCS  IBM MF United Kingdom
      CP00297   E SBCS  IBM MF France
      ```

      For a description of all of the fields in CPXCPTBL, see *Structure of the Known Code Page File (CPXCPTBL)* on page 307.

   b. Copy the lines for the additional or alternate code pages you need from member CPXCPTBL to member CPCODEPG.

      For example, to add French code page 297 to your configuration, copy the following line from CPXCPTBL to CPCODEPG:

      ```
      CP00297   E SBCS  IBM MF France
      ```

      If you started with the default CPCODEPG FILE, the new version will contain the following information:

      ```
      CP00037   E SBCS  US IBM MF EBCDIC cod
      CP00437   A SBCS  US PC ASCII code
      CP00137   A SBCS  ANSI Character Set for MS-Windows
      CP01047   E SBCS  IBM MF Open Systems (Latin 1)
      CP65001   A UTF8  Unicode (UTF-8)
      CP00297   E SBCS  IBM MF France
      ```

Note: If the desired code page is not listed in the known code page file (CPXCPTBL), refer to the appropriate IBM Character Data Representation Architecture (CDRA) document and create your own, or contact your local TIBCO representative for information about additional code pages.

4. Execute the FOCUSC CLIST to start FOCUS.

   The ISETUP installation procedure creates a CLIST named FOCUSC in the *install_hlq*.CONF.DATA library. This CLIST allocates all of the required libraries for executing FOCUS using the high-level qualifier you chose during the installation process.

   Alternatively, you can submit the FOCUS JCL generated by the installation process, member FOCUS in the *install_hlq*.CONF.DATA library. In this case, place the TSGU command in SYSIN. For an example, see *How to Use the TSGU Command* on page 308.

5. At the FOCUS command prompt, issue the TSGU command to generate the necessary tables.

   For example, to generate the transcoding table (TRANTBL) that adds the new or alternate code pages from the CPCODEPG file, issue the following command.

   ```
   TSGU
   ```

   The tables are created as members in the *install_hlq*.CONF.CFG library.

   Next, configure the FOCUS default code page and language as described in *Configuring FOCUS To Use a New Code Page* on page 299.

If you modified the monocasing information in any of the code page definition files, issue the following command to generate the monocasing table (CASETBL):

```
TSGU CASE
```

If you modified the sorting sequence in any of the code page definition files, issue the following command to generate the sorting table (SORTTBL):

```
TSGU SORT
```

To create all three types of tables (transcoding, monocasing, and sort tables), issue the following command:

```
TSGU GEN
```

After running the TSGU, check the *install_hlq*.CONF.CFG library to make sure the required tables were generated.

For complete information about TSGU command syntax, see *Using the TSGU to Generate New Transcoding Tables* on page 308.

Configuring FOCUS To Use a New Code Page

When you invoke FOCUS, one language and code page combination will be in effect by default. When you configure NLS Services, the default language and code page are controlled by settings in the NLSCFG configuration file, member NLSCFG in the *install_hlq*.CONF.CFG library.

For example, to change the default language to French and the code page to 297, enter the following settings in NLSCFG:

```
LANG=FRE
CODE_PAGE=297
```

If there is no NLSCFG member in the concatenation of data sets allocated to DDNAME ERRORS, the FOCUS default language and code page will be in effect. For information, see *The FOCUS Default Code Page Configuration* on page 295.

For complete information about the NLSCFG configuration file, see *How to Use the NLS Configuration File (NLSCFG)* on page 304.

Configuring PC3270 Session Parameters

Before invoking FOCUS, you must configure PC3270 to use the same code page that will be used by FOCUS.

1. Log off your z/OS session.
2. On the PC3270 Communication menu, select *Configure*.
3. Click *Session Parameters*.
4. Select the code page that FOCUS will use from the list of code pages, and click *OK*.

PC3270 will display a message about the new configuration. Accept the change to the new PC3270 code page, and wait for the screen to refresh before logging in again.

Verifying the FOCUS Language Configuration

Once you have configured FOCUS for a new code page and language, you should verify that they are in effect when FOCUS is invoked.

1. Log in to z/OS and invoke FOCUS. You can use the FOCUSC CLIST that was created by the installation procedure.
2. Issue the ? LANG command to check that the language and code page you configured are in effect. For example:

```
          NATIONAL LANGUAGE INFORMATION

Language                 033/FRENCH      (FRE,fr)
Code Page                297
Client Code Page         297
Dollar                   5B($)
Lowcase alphabet         YES
Decimal notation         OFF(.)
Currency symbol          $
Date/Time format         EDA
NLS sort                 NO(BINARY)
NLS upcase/lowcase       NO
NLS Control Characters   OFF
DBCS Flag                OFF(SBCS)
```

3. Display data that contains characters specific to the new code page to make sure that they display correctly.

Advanced NLS Configuration Options

This section describes how to customize monocasing and sort sequences. Although these advanced options are available, they are not needed in a typical NLS configuration.

Configuring Customized FOCUS NLS Monocasing

Monocasing (also called Case Conversion) is the conversion of a letter from its lowercase to uppercase form (or vice versa). As part of the basic FOCUS initialization, FOCUS is set up with standard monocasing where all requests, except for data between single quotes, are converted to uppercase according to the monocasing table (CASETBL). The monocasing table file *install_hlq*.CONF.CFG(CASETBL) converts a-z to A-Z only. If you require customized monocasing, such as special upper/lowercase accented characters, then you must modify the code page definition file (CP*nnnnn*) and then generate a new NLS monocasing table file (CASETBL) using the TSGU. The new monocasing table is based on the changes made to the code page definition file (CP*nnnnn*).

Note: The TIBCO FOCUS functions LOCASE and UPCASE respect the NLS monocasing table file (CASETBL).

Procedure: How to Customize Your NLS Monocasing Table

Note: As part of the basic initialization, monocasing tables are provided for most of the common European languages. You will only need to customize the monocasing tables if you require a special monocasing configuration.

NLS monocasing involves language-sensitive (code page sensitive) uppercase and lowercase conversion. You can customize the attributes of each character by completing the following steps:

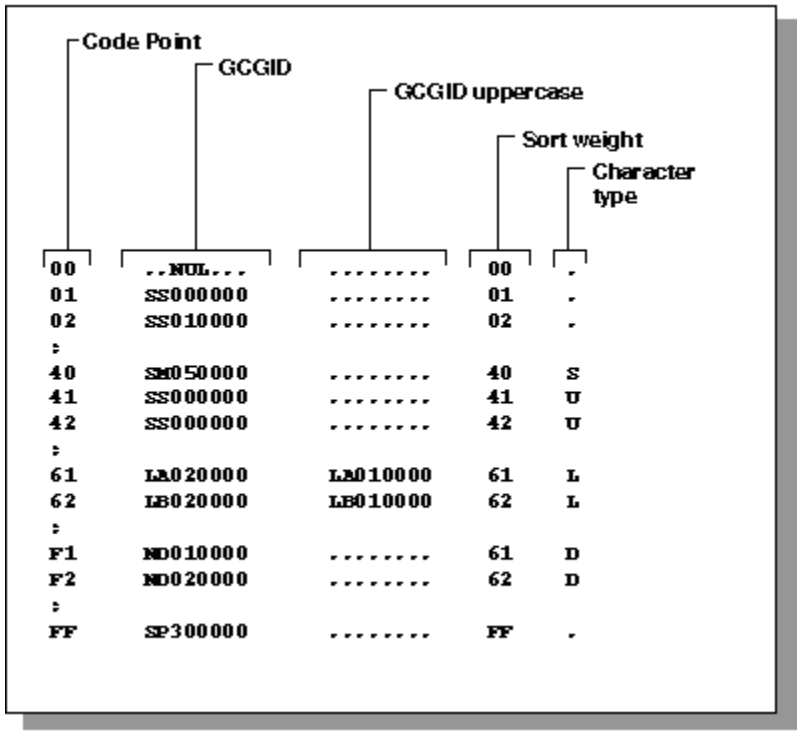
1. Copy the code page definition file you want to edit from the *install_hlq*.ERRNLS.DATA library to the *install_hlq*.CONF.CFG library. The code page definition file (CP*nnnnn*) is named by the code page number. For example, CP00037 contains the monocasing information for US English code page 37. For more information on the code page definition file (CP*nnnnn*), see *How to Use the Code Page Definition File (CPnnnnn)* on page 307.

   Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.

2. Edit the code page definition file.

   The code page definition file (CP*nnnnn*) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to GCGID uppercase in the third column and Character type in the fifth column.

   The following is a chart of a sample code page definition file layout:



3. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Adding New or Alternate Code Pages for FOCUS* on page 296. The updated code page generation list (CPCODEPG) is used to regenerate the custom monocasing table file (CASETBL).
4. Execute the TSGU with the parameter CASE.

   The TSGU generates the updated NLS monocasing table file *install_hlq*.CONF.CFG(CASETBL).

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library or contact your local TIBCO representative.

Configuring Customized FOCUS NLS Sort Sequences

As part of the basic FOCUS initialization, FOCUS is set up with standard sorting where FOCUS uses sort sequences of the binary representation of a character string. If you require customized sorting, such as changing your sort order to account for Swedish umlauts (Ü), then you must modify the NLS sorting table file (SORTTBL).

Procedure: How to Customize Your NLS Sort Tables

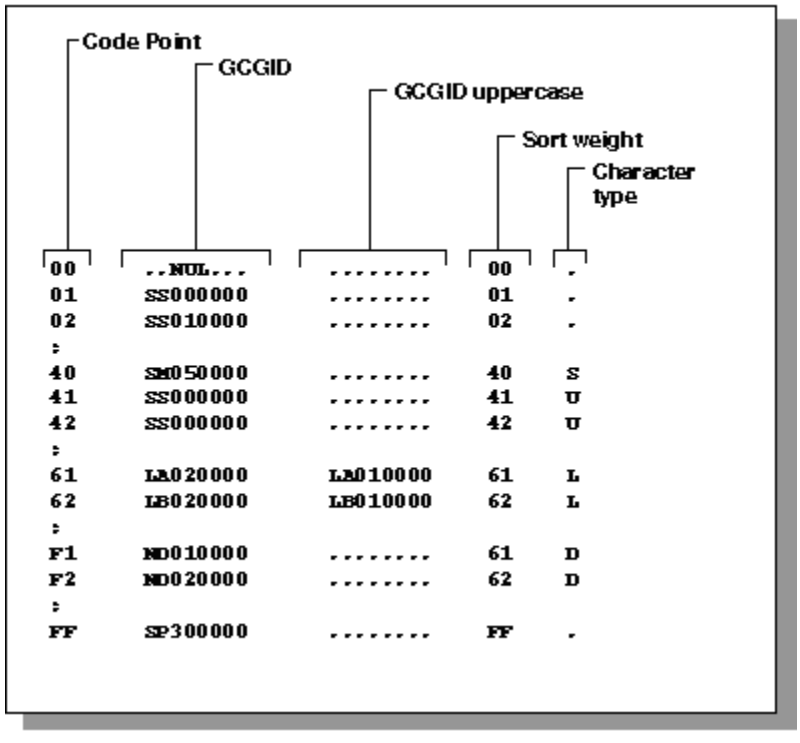
Note: As part of the basic initialization, sorting tables are provided for most of the common European languages. You will only need to customize the sorting tables if you require a special sorting sequence.

If you want to use a weighted sort that accounts for characters that are out of binary sequence, you can customize the sort tables by completing the following steps.

1. Copy the code page definition file *install_hlq*.ERRNLS.DATA(CP*nnnnn*), which is named by the code page number, to *install_hlq*.CONF.CFG(CP*nnnnn*). For example, CP00037 contains the sorting information for US English code page 37.

   Note: You can reference the known code page file (CPXCPTBL) to find the name of the code page definition file.
2. Edit the code page definition file For more information on the code page definition file (CP*nnnnn*), see *How to Use the Code Page Definition File (CPnnnnn)* on page 307.

The code page definition file (CP*nnnnn*) contains the Code Point and Graphic Character Global Identifier (GCGID). Make the appropriate changes to the Sort weight listed in the fourth column. The following is a chart of a sample code page definition file layout:



3. Edit the code page generation list file (CPCODEPG) and add the code page definition information as described in *Adding New or Alternate Code Pages for FOCUS* on page 296. The updated code page generation list (CPCODEPG) is used to regenerate the custom sorting table file (SORTTBL).
4. Execute the TSGU from FOCUS with the parameter SORT.

   The TSGU generates the updated NLS sorting table file *install_hlq*.CONF.CFG(SORTTBL).

Note: For additional information on modifying monocasing values in the code page definition file, refer to the IBM CDRA Library or contact your local TIBCO representative.

Using the NLS Configuration Files

This section provides detailed descriptions of the NLSCFG configuration file, the Code Page Definition File (CP*nnnnn*), and the Known Code Page File (CPXCPTBL).

Syntax: How to Use the NLS Configuration File (NLSCFG)

The NLSCFG configuration file controls the code page and the language used for error messages and characters in FOCUS. You can configure several different code pages and languages in the CPCODEPG file. Using the NLSCFG file, you can specify which one will be in effect when you invoke FOCUS.

You change the FOCUS code page and language by entering LANG and CODE_PAGE settings in the NLSCFG configuration file. Changing the LANG setting sets all other parameters to valid values.

Note: The LANG setting should match the three-character language abbreviation (language ID) or the language name.

You can find the appropriate language abbreviation for each code page in member CPCPALL in the alias library *install_hlq*.ERRNLS.DATA. The following shows a portion of the CPCPALL file, which includes the language setting for code page 297:

```
* LANG=FRE
cp297
cp863
cp137
cp1047
cp37
cp65001
BR
* LANG=GER
cp273
cp850
cp137
cp1047
cp37
cp65001
BR
```

If you do not configure FOCUS for NLS services, the LANG value used by FOCUS is AMENGLISH. To change the default language to French, enter the following setting in the NLSCFG configuration file:

```
LANG = FRE
```

Error messages are only translated for German, Spanish, French, Dutch, and Japanese. Error messages for all other languages are translated to English.

Note: Swedish and some other languages may only be partially translated.

The following is a reference of keywords in the NLSCFG file and their associated values:

LANG=

Is the language. The list of known languages file (CPCPALL), located in *install_hlq*.ERRNLS.DATA, contains a list of valid values. The value of LANG determines the defaults for other parameters in FOCUS. The default value is AMENGLISH (American English).

COLLATION=

Establishes the collation (sort) sequence. Valid values are:

- **CODEPAGE**. Bases collation sequence on the code page in effect. This is the default value.
- **BINARY.** Bases the collation sequence on binary values. When code page files are not modified, CODEPAGE is the same as BINARY, except when LANGUAGE is Danish, Finnish, German, Norwegian or Swedish in an EBCDIC environment.
- **SRV_CS.** Bases collation sequence on the LANGUAGE setting, and is case-sensitive.
- **SRV_CI.** Bases collation sequence on the LANGUAGE setting, and is case-insensitive

This setting can be overridden by the SET COLLATION command.

DATEOUTPUT=

Localizes month and week names on date format output for display options T, t, TR, tr, W, w, WR and wr. Valid values are:

- **DEFAULT.** Generates English month and week names regardless of the language setting. This is the default setting.
- **LOCALIZED.** Generates localized month and week names for the current language setting.

  For example, assume FOCUS is configured to use French code page 297. In the following request, the HIRE_DATE field is displayed with the YYMDtr display option:

```
TABLE FILE EMPLOYEE
SUM CURR_SAL HIRE_DATE/YYMDtr
BY DEPARTMENT
ON TABLE SET PAGE NOPAGE
END
```

Omitting the DATEOUTPUT setting from NLSCFG or entering DATEOUTPUT=DEFAULT displays month names in English:

```
DEPARTMENT          CURR_SAL  HIRE_DATE
----------          --------  ---------
MIS              $108,002.00  1981, November  2
PRODUCTION       $114,282.00  1982, February  2
```

Entering the setting DATEOUTPUT=LOCALIZED in NLSCFG translates month names to French:

```
DEPARTMENT          CURR_SAL  HIRE_DATE
----------          --------  ---------
MIS              $108,002.00  1981, novembre  2
PRODUCTION       $114,282.00  1982, février  2
```

CURRENCY=

Is a one-to-four character string or a hexadecimal value that identifies a currency symbol. You can use any characters. The default currency symbol is the dollar sign ($). Examples of popular currency symbols and their corresponding character abbreviations include the following.

- **€** or **EUR** for euro.
- **¥** or **JPY** for yen.
- **$** or **USD** for U.S. dollar. This is the default currency symbol.
- **£** or **GBP** for British pound sterling.
- **NIS** for Israeli New Shekel.

The following are examples of hexadecimal values for code page 137.

- **0x80** for the euro symbol.
- **0x24** for the dollar sign.

Example: Sample NLSCFG File

The following NLSCFG file specifies French as the language and 297 as the code page.

```
LANG=FRE
CODE_PAGE = 297
DATEOUTPUT=DEFAULT
COLLATION=CODEPAGE
```

Syntax: How to Use the Code Page Definition File (CP*nnnnn*)

The code page definition file *install_hlq*.ERRNLS.DATA(CP*nnnnn*) contains information on the characters for each code point value in the code page.

```
dd aaaaaaaa aaaaaaaa xx h
```

where:

dd

Is the hexadecimal code point value (00 through FF).

aaaaaaaa

Is the Graphic Character Global IDentifier (GCGID).

xx

Is the Sort weight. Consists of a hexadecimal code point value (00 through FF).

h

Is the character type. Possible values are:

L for Lower-case alphabet.

U for Upper-case alphabet.

A for Asian (non-alphabet) character.

D for Digit.

S for Special character.

C for Control (non-printable) character.

Reference: Structure of the Known Code Page File (CPXCPTBL)

When you add code pages to the FOCUS configuration, you copy code page information from the known code page file, member CPXCPTBL in the *install_hlq*.ERRNLS.DATA library.

```
CPnnnnn b dbcs-id description
```

where:

CP

Is the code page prefix (always CP).

nnnnn

Is the code page number.

b

Is the character type. Possible values are:

A for ASCII.

E for EBCDIC.

dbcs-id

Is the DBCS identifier (DBCSID).

description

Is a description of the code page.

Using the TSGU to Generate New Transcoding Tables

The TSGU is a multi-functional utility that is used to create the transcoding table (TRANTBL), sorting table (SORTTBL), and monocasing table (CASETBL).

Syntax: How to Use the TSGU Command

```
TSGU
TSGU GEN
TSGU KTBL {pdfy_name|??}
TSGU INFO info-command
TSGU CASE
TSGU SORT
```

where:

no parameters

Generates the NLS transcoding table (TRANTBL).

GEN

Generates all the tables (TRANTBL, CASETBL and SORTTBL).

INFO

Displays information about the current NLS configuration.

CASE

Generates the custom monocasing table (CASETBL) binary file.

SORT

Generates the custom sort sequence table (SORTTBL) binary file.

```
KTBL PD??
```

Generates the special font information tables (PDFYTBL) when Unicode fonts should be used in PDF format files. You can specify a *pdfy-name* font metrics file for a specific font, or use PD?? to generate all of the files.

Example: Submitting JCL to Generate the Transcoding Table and Unicode PDF Font Tables

In this example, the high-level qualifier for your FOCUS production libraries is FOC7706.

Taiwanese code page 937 has been added to member CPCODEPG in FOC7706.CONF.CFG:

```
CP00037   E SBCS  US IBM MF EBCDIC code
CP00437   A SBCS  US PC ASCII code
CP00137   A SBCS  ANSI Character Set for MS-Windows
CP01047   E SBCS  IBM MF Open Systems (Latin 1)
CP65001   A UTF8  Unicode (UTF-8)
CP00937   E SOSI  Taiwanese IBM MF (cp37+cp835)
```

The following JCL invokes FOCUS, issues the TSGU commands to generate the transcoding and PDF tables, and exits FOCUS:

```
//*  YOUR JOB CARD GOES HERE
//*
//FOCUS    EXEC PGM=FOCUS
//STEPLIB  DD DSN=FOC7706.FOCLIB.LOAD,DISP=SHR
//ERRORS   DD DSN=FOC7706.CONF.CFG,DISP=SHR
//         DD DSN=FOC7706.ERRORS.DATA,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
TSGU
TSGU KTBL PD??
FIN
/*
```

SYSPRINT will show various Generating File messages.

Member NLSCFG in FOC7706.CONF.CFG has been created with the following commands to invoke FOCUS with the new code page and language:

```
LANG = T-CHINESE
CODE_PAGE = 937
DATEOUTPUT = DEFAULT
COLLATION = CODEPAGE
```

Now, you can invoke FOCUS, and code page 937 will be in effect with Taiwanese as the configured language. Issue the following command so that your emulator can display the Chinese characters:

```
SET TERMINAL=IBM5550
```

Syntax: How to Use the TSGU Info-Commands

```
TSGU INFO CP [DBCS] [nnn [nnn [...]]] [MAT [DES]]
TSGU INFO CP [SBCS] [nnn [nnn [...]]] [MAT [DES]]
TSGU INFO TRAN [idx [idx [...]]] [M] [V]
TSGU INFO CASE [nnn [nnn [...]]] [M] [V]
TSGU INFO SORT [nnn [nnn [...]]] [M] [V]
TSGU INFO SET
```

where:

`CP`

Lists transcoding tables.

`DBCS`

Lists only DBCS transcodings.

`SBCS`

Lists only SBCS transcodings.

`nnn`

Code page which only shows given code page transcodings (no leading zeros necessary).

`idx`

Index which only shows given index transcodings.

`MAT`

Creates a matrix report for transcoding tables.

`DES`

Reverses an order of code pages in a matrix report.

`TRAN`

Shows transcoding table contents (trantbl.err).

`CASE`

Shows uppercase/lowercase table contents (casetbl.err).

`SORT`

Shows sort table contents (sorttbl.err).

`M`

Masks null transcode values in tables.

```
V
```

Shows tables in vertical layout.

```
SET
```

Shows the list of available languages.

Syntax: How to Use the TSGU ? Command

To see the online help information for the TSGU command, issue the following command:

```
TSGU ?
```

The output is:

```
TSGU  : GEN   : cpcodepg-id ::
      : TRAN  : cpcodepg-id :: (default)
      : CASE  : cpcodepg-id ::
      : SORT  : cpcodepg-id ::
      : INFO  : info-command ::
      : XLAT  : infile  : outfile  :::
      : XLAT  TEMPLATE : cp1 cp2 ::
      : KTBL  : ktbl-command : : xxxx : xxxx : ... ::::
      : ERR   err-command :

 GEN  : generate all necessary tables (binary files)
 TRAN : generate a TRANTBL and KTBL binary files
 CASE : generate a CASETBL binary file
 SORT : generate a SORTTBL binary file
 INFO : display info of the current Translation Services (? LANG)
 XLAT : convert a file by specified Code Pages in config file (tsgu.nls)
       TEMPLATE shows a sample batch file/unix shell script for multi-
       files to be XLATed. (tsgult only)
 KTBL : generate individual KTBL binary files
 ERR  : convert EDAHOME/nls/*.err file (UNIX and Windows NT only)
 xxxx : DBCS translation table identifier

cpcodepg-id:
    Max 4 characters. To use customized and saved CPCODEPG.NLS
    If specified, a file named 'cpcp:cpcodepg-id:.nls' is used instead.
    cpcpall.nls and cpcpucs.nls are provided as samples.
```

```
info-command:
    CP   :| DBCS |: : nnn | nnn | ... : : MAT : DES ::
          | SBCS |
    TRAN : idx : idx : ... :: : M : : V ::
    CASE : nnn : nnn : ... :: : M : : V ::
    SORT : nnn : nnn : ... :: : M : : V ::
    ALL

    CP   : Show TRTIDX
    DBCS : Show only DBCS translations
    SBCS : Show only SBCS translations
      idx  : Index, show only given indexes translations
      nnn  : Code Page, show only given Code Pages translations
      MAT  : Make a matrix report for TRTIDX
      DES  : Reverses an order of Code Pages in a matrix report
      TRAN : Show translation tables (TRANTBL)
      CASE : Show upcase/lowcase tables (CASETBL)
      SORT : Show sort tables (SORTTBL)
      M    : Mask null translate values in tables
      V    : Show tables vertical layout against horizontal
      ALL  : Show intlcm info for debugging

ktbl-command:
      CHK
      FMT
      CNV : UDC : : nn | CPU | MEM :
      RVS : UDC : : nn | CPU | MEM :
      LST : FFFF :
      ADD

      CHK : Check the KTBLs translation integrity
      FMT : Format the KTBLs
      CNV : Convert the KTBLs
      RVS : Make a reverse translation table for the KTBLs
      LST : List the DBCS translation
      ADD : Make a new DBCS translation from two KTBLs
      UDC : Remove Unused UDC code translation
      nn  : Threshold level for indexing the ranges
      CPU : nn = 65536, to make only one index range
      MEM : nn = 4, to make the smallest size format for the KTBLs
      FFFF: Produce entries for source code even they are translated
             to invalid codes

err-command:
     GEN  aaa
     UCS  aaa

     GEN : Create *:aaa:.err files in the directory EDACONF/etc.
           (JPE and GE5 only)
     UCS : Create *(aaa).err files in Unicode (utf-8) into the
           directory EDACONF/etc.
     aaa : Language abbreviation
```

Chapter 10

Unicode Support

Unicode is a universal character encoding standard that assigns a code to every character and symbol in every language in the world. Since no other encoding standard supports all languages, Unicode is the only encoding standard that ensures that you can retrieve or combine data using any combination of languages.

Note: In order to use Unicode in a z/OS environment, you must add code page 65002 to your configuration. For more information about configuring National Language services, see *Configuring FOCUS for National Language Support Services* on page 291.

In this chapter:

- Unicode Encoding Standards
- Accessing Unicode Data
- Selecting, Reformatting, and Manipulating Characters
- Sort Order Under Unicode
- Added Unicode Support for Master Files, Data Files, and Application Directory Names
- Unicode PDF Output

Unicode Encoding Standards

FOCUS supports a Unicode Transformation Format (UTF) called UTF-8 in ASCII environments, and UTF-EBCDIC in EBCDIC environments:

- **For ASCII,** the UTF-8 encoding standard assigns each character of each language a code that can be from 1-3 bytes long. The codes assigned to European characters are 1 or 2 bytes long, Middle Eastern characters are 2 bytes long, and those assigned to Asian characters are 3 bytes long. This standard is compatible with ASCII format because the first 128 UTF-8 codes have the same 1-byte representation as the corresponding ASCII codes.
- **For EBCDIC,** the UTF-EBCDIC encoding standard assigns each character a code that can be from 1-4 bytes long. EBCDIC characters, including C1 control characters, have the same 1-byte representation in UTF-EBCDIC.

In non-Unicode single-byte encoding standards, such as ASCII, each character is assigned a code that is 1 byte long, limiting the number of characters that can be represented by the standard. When using those standards, it became common to equate a character with a byte of storage. If you had a string of 10 characters, the amount of storage needed was 10 bytes, and many character manipulation routines expected character string lengths to be specified as a number of bytes.

With Unicode encoding, bytes and characters are no longer equated. Characters are represented internally by a varying number of bytes, depending on the character. If you configure FOCUS for Unicode, you define the length of strings and alphanumeric fields in terms of characters, not bytes. This simplifies specifying string and field lengths. Each character is represented internally by up to 3 bytes (4 for EBCDIC Unicode), and FOCUS automatically adjusts for the actual storage length. In reports, each character displays in a report column using one space, regardless of how many bytes it takes up in memory. This character-based processing mode employed for Unicode environments is called *character semantics*. The non-Unicode mode is called *byte semantics*.

Procedures that had been developed using byte semantics will continue to work when deployed in a Unicode environment, without adjustment, in most cases.

To compress trailing blanks and display columns as the width of the largest actual data value, issue the SET SQUEEZE=ON command.

The main benefit of the new system is the ability to have multiple languages (both European and Asian) in the following FOCUS and Dialogue Manager objects:

- Titles, descriptions, and names in synonyms.
- Headings and prompts in procedures.
- Data for all supported adapters (for example, SAP BW, SAP R/3-ECC, Oracle, DB2, Sybase ASE, Sybase IQ, Teradata, MySQL, Web Services, fixed-format sequential files). For more information, see *Accessing Unicode Data* on page 315.

For information about configuring FOCUS for Unicode, see the chapter on *Configuring FOCUS for National Language Support Services* on page 291.

Accessing Unicode Data

Adapters for the following types of data sources support Unicode:

- DB2
- Fixed-format sequential files
- Microsoft SQL Server
- MySQL
- Oracle
- SAP BW
- SAP R/3-ECC
- Sybase ASE
- Sybase IQ
- Teradata (CLI)
- Web Services
- XBRL
- XML

Relational adapters in a Unicode environment assume that the DBMS returns character data to FOCUS already converted to Unicode. The relational adapters convert data to the correct DBMS API when writing to a relational data source (for example, Oracle to UTF-8, Microsoft SQL Server to UTF-16, and DB2 on z/OS to UTF-EBCDIC).

XML-based adapters (the Adapter for XML, and the Adapter for XBRL) obtain the code page from the XML declaration of the processed XML document. For more information, see *http://www.w3.org/TR/REC-xml#sec-prolog-dtd*.

The Adapter for Web Services generates SOAP requests using the UTF-8 code page.

Reference: **Unicode Considerations for Oracle**

The adapter supports Unicode data in Oracle release 10g or higher databases that have been configured with the NLS_CHARACTERSET parameter set to UTF8. You must set the NLS_LANG environment variable in the edastart file, in a separate shell file, in a database profile, or in a user profile.

Set NLS_LANG using the following syntax

```
NLS_LANG = language_territory.characterset
```

where:

language

Is the selected language.

territory

Is the name of the country associated with the selected language.

characterset

Is the value of the NLS_CHARACTERSET variable that is set in the Oracle database. For Unicode, this is always UTF8.

For example, for American English UTF-8, you would use the following setting:

```
NLS_LANG=American_America.UTF8
```

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: Unicode Considerations for DB2

Information Builders supports DB2 databases, version 8 and higher. To prepare the DB2 environment for Unicode on:

- **Windows,** the database must have been created with the option CODESET UTF-8, and you must add the following variable to the environment using Windows or in the edastart file:

  ```
  DB2CODEPAGE=1208
  ```

- **UNIX,** the database must have been created with the option CODESET UTF-8, and you must set the LANG and NLS_LANG environment variables in the edastart file or in a separate shell file.

  For example, for American English, you would export the following variables:

  ```
  export LANG=EN_US.UTF-8
  export NLS_LANG=American_America.UTF8
  ```

- **z/OS,** the database must have been created with the CCSID UNICODE option, and you must ensure that the DSNAOINI environment variable points to a configuration file containing the following specification:

  ```
  CURRENTAPPENSCH=UNICODE
  ```

  The adapter supports Unicode only with the CLI interface.

  In a Unicode environment, the Adapter for DB2 requires a BIND command for PREPARE/EXECUTE logic using parameter markers.

  For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

- **IBM i**, Unicode is only supported in DB2 CLI mode. DB2 allows column-by-column specification of CCSID information at CREATE TABLE time. The columns may or may not be explicitly Unicode 1208 and 1200, however, all CCSIDs are transcoded to FOCUS as CSID 1208 UTF8. Thus, any existing table may be used in a Unicode configuration regardless of its underlying CCSID specifications.

Reference: Unicode Considerations for Sybase ASE

The adapter supports Unicode data in Sybase ASE version 15.0 and higher databases that have been created with the CHARACTER SET option set to UTF-8. You must set the LANG and NLS_LANG environment variables in the edastart file or in a separate shell file before starting FOCUS.

For example, for American English, you would export the following variables:

```
export LANG=EN_US.UTF-8
export NLS_LANG=American_America.UTF8
```

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: Unicode Considerations for Sybase IQ

Beginning with Sybase IQ version 12.7, the adapter supports Unicode data in Sybase IQ databases that have been created with the UTF-8 character set. You must set the LANG and NLS_LANG environment variables in the edastart file or in a separate shell file before starting FOCUS.

For example, for American English, you would export the following variables:

```
export LANG=EN_US.UTF-8
export NLS_LANG=American_America.UTF8
```

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: Unicode Considerations for Microsoft SQL Server

The adapter, using the OLE DB interface, supports Unicode data stored in NCHAR and NVARCHAR fields (where N stands for national). N columns can support data of any language or combination of languages.

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: Unicode Considerations for Teradata (CLI)

The Adapter for Teradata (CLI) supports Unicode UTF-8 format if:

- The Teradata CLI client components are part of release TTU8.0 or higher.
- The Teradata database is release V2R6.0 or higher and appropriate language support was enabled during the sysinit process.

Contact your database administrator (DBA) to determine whether international language support has been enabled in your Teradata system and/or consult the Teradata documentation for details about International Character Set support.

Note that, at the present time, when Unicode is enabled the length of a Teradata Column Name and/or TITLE cannot exceed 21 characters (bytes).

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: Unicode Considerations for MySQL

The Adapter for MySQL is implemented using JDBC. This implementation supports Unicode data stored in character fields with CHARACTER SET set to UTF-8.

You must set the LANG environment variable in the edastart file or in a separate shell file before you start FOCUS. For example, for American English you would export the following variable:

```
export LANG=EN_US.UTF-8
```

For information about data type support, see *Relational Adapter Data Type Support for Unicode* on page 319.

Reference: **Relational Adapter Data Type Support for Unicode**

In Unicode databases the information in CHAR(n) columns is stored in a UTF-8 encoding scheme. Most RDBMS Unicode columns of CHAR type specify length in bytes, not characters. The B-modifier in the Actual format denotes that a character column with a fixed byte length might contain a varying number of UTF-8 characters. This is reflected in the AnV Usage format.

DBMS	Column Type	Usage	Actual*
DB2	CHAR(*n*)	A*n*V	A*n*B
	GRAPHIC(*n*)	A*n*	A*n*
	VARCHAR(*n*)	A*n*V	A*n*VB
	VARGRAPHIC(*n*)	A*n*V	A*n*V
Microsoft SQL Server	CHAR(*n*) single byte code page	A*n*	A*n*
	CHAR(*n*) double byte code page	A*n*V	A*n*V
	NCHAR(*n*)	A*n*	A*n*
	VARCHAR(*n*)	A*n*V	A*n*V
	NVARCHAR(*n*)	A*n*V	A*n*V
MySQL	CHAR(*n*)	A*n*	A*n*
	VARCHAR (*n*)	A*n*V	A*n*V
Oracle	CHAR(n CHAR)	A*n*	A*n*
	CHAR(n BYTE)	A*n*V	A*n*B
	NCHAR(*n*)	A*n*	A*n*
	VARCHAR(*n* CHAR)	A*n*V	A*n*V
	VARCHAR(*n* BYTE)	A*n*V	A*n*VB
	NVARCHAR(*n*)	A*n*V	A*n*V

DBMS	Column Type	Usage	Actual*
Sybase ASE	CHAR(*n*)	A*n*	A*n*B
	UNICHAR(*n*)	A*n*	A*n*
	VARCHAR(n)	A*n*V	A*n*VB
	UNIVARCHAR(*n*)	A*n*V	A*n*V
Sybase IQ **	CHAR(*n*)	A*n*	A*n*B
	VARCHAR(n)	A*n*V	A*n*VB
Teradata	CHAR(*n*)	A*n*	A*n*
	VARCHAR (*n*)	A*n*V	A*n*V

* Note that on EBCDIC platform(s) the ACTUAL size for a B-suffixed format is increased 1.5 times to accommodate the expansion when converting from UTF-8 to UTF-EBCDIC. For example, on z/OS the synonym created for a DB2 CHAR(10) column contains the following: USAGE=A10, ACTUAL=A15B.

****Note the following limitation for Sybase IQ:** You cannot use the HOLD FORMAT SYBASE command with the Unicode implementation of Sybase IQ since this command depends on the availability of UNICHAR and UNIVARCHAR data types, which are not supported by Sybase IQ.

Reference: **Unicode Considerations for SAP BW and SAP R/3-ECC**

SAP uses UTF-16 encoding in its Unicode system. FOCUS uses UTF-8 and handles all conversions between the two encoding schemes. FOCUS may not need to be configured for Unicode when accessing the SAP Unicode system.

NLS settings for FOCUS must be configured in such a way that the FOCUS code page can handle the list of chosen languages. For example, ISO 8859-1 can accommodate most Western European languages. The 8859 family can handle character specifics with the lower set almost being mapped to US ASCII. Therefore, with 8859-1 one could request English, German, French, and Spanish. When a character set requires a code page that takes more than one byte per character (for example, many Asian languages), the only choice for FOCUS is 65001 (UTF-8).

The adapters provide access to Unicode SAP BW and SAP ECC systems, respectively. This extends support of data and metadata in multiple languages to FOCUS, consistent with support by the SAP server. A synonym can be created using one or more languages. Those languages will be used to create titles and descriptions.

- For SAP BW, the userid and password in the sapserv.cfg file must be able to connect to SAP BW using the enumeration of desired languages.
- For SAP R/3-ECC, FOCUS logon language is used to retrieve all languages.

Reference: **Unicode Considerations for Fixed-Format Sequential Files**

When retrieving a fixed-format sequential file, FOCUS attempts to determine the code page the file was meant to be retrieved with by checking the Master File CODEPAGE attribute. If the Master File does not contain the CODEPAGE attribute, FOCUS code page is used to read the file.

In a Unicode configuration, HOLD files in BINARY and ALPHA formats are created using UTF-8 conversion, which assigns each character three bytes of storage in ASCII environments or four bytes in EBCDIC environments. Fields defined in the Master File using the data type A in both the USAGE and ACTUAL attributes are described in terms of characters. Fields defined using any other combination of USAGE and ACTUAL attribute values are described in terms of bytes.

To force a field in a fixed-format sequential file to be described in terms of bytes, add B to the end of the ACTUAL attribute. For example, to specify that a field is stored in 10 bytes, you would specify:

```
ACTUAL=A10B
```

The adapter will then read the specified number of bytes from the record and convert their contents to the number of characters specified by the file code page.

Regardless of how much storage a character occupies, it occupies only one space on a report, as always.

Selecting, Reformatting, and Manipulating Characters

In character semantics mode, selection tests against a mask are automatically adjusted to work with characters rather than bytes. Formats assigned by reformatting a field in a request or by defining a temporary field are interpreted in terms of characters. Character functions interpret all lengths in terms of characters.

Example: Defining a Virtual Field

Consider the following DEFINE in the Master File for the EMPLOYEE data source:

```
DEFINE FIRST_ABBREV/A5 WITH FIRST_NAME = EDIT(FIRST_NAME, '99999$$$$$');$
```

In character semantics mode, format A5 is interpreted as five characters (up to 15 bytes on ASCII platforms, up to 20 bytes on EBCDIC platforms), and the comparison is performed based on this number of bytes. In byte semantics mode, format A5 is interpreted as five bytes, and the comparison is performed based on five bytes. In either case, the correct characters are compared and extracted.

Example: Reformatting a Field

Consider the following PRINT command:

```
PRINT FIELD1/A10
```

In character semantics mode, format A10 is interpreted as 10 characters (up to 30 bytes), meaning that up to 30 bytes must be retrieved when this field is referenced. In byte semantics mode, format A10 means that 10 bytes will be retrieved. In either case, the field displays as 10 characters that take up 10 spaces on the report output.

Reference: Character Functions That Support Character Semantics

In character semantics mode, all character manipulation functions interpret lengths in terms of characters. The following functions operate on alphanumeric strings in character semantics mode when Unicode is configured:

- **String manipulation and extraction functions.**

  GETTOK, OVRLAY, PARAG, REVERSE, SQUEEZ, STRIP, SUBSTR, SUBSTV, TRIM, TRIMV

- **Justification functions.**

  CTRFLD, LJUST, RJUST

- **Length and position functions.**

  ARGLEN, LENV, POSIT, POSITV

- **Format conversion functions.**

  EDIT

- **Decoding, comparison, and editing functions.**

  CHKFMT, EDIT, DECODE, SOUNDEX

- **String replacement functions.**

  CTRAN, HEXBYT, BYTVAL (see notes below), STRREP

- **Case translation functions.**

  LCWORD, LOCASE, LOCASV, UPCASE, UPCASV

Note: The HEXBYT, BYTVAL, and CTRAN functions have been extended to handle multibyte characters in Unicode configurations. These functions use or produce numeric values to represent characters. In Unicode configurations, they use or produce values in the range:

- 0 to 255 for 1-byte characters
- 256 to 65535 for 2-byte characters
- 65536 to 16777215 for 3-byte characters
- 16777216 to 4294967295 for 4-byte characters (primarily for EBCDIC)

To find the numeric value corresponding to a given character, find its hexadecimal code and convert to decimal with a hex calculator such as the Windows XP Calculator program. (Make sure to use the UTF-8 or UTF-EBCDIC code, not the Unicode code point, which would be the UTF-16 value.)

For example, assume you would like to create a variable of format A1 containing the euro sign. The euro sign in UTF-8 is, in hex, E282AC. Converting this to decimal gives 14849492. Thus, the proper DEFINE or COMPUTE would be:

```
EUROSIGN/A1 = HEXBYT(14849492, 'A1');
```

If you are creating a FOCEXEC with a UTF-8 compliant editor, you can also get the value of the euro sign in this way:

```
EUROVAL/I8 = BYTVAL('€', 'I8');
```

The CTRAN function replaces all occurrences of a character in a string with another character, given the decimal values that represent the hexadecimal codes for the two characters. Traditionally, this technique was used to replace characters that were difficult to input directly. Decimal values of characters can be complicated to determine. Therefore, if you want to replace characters or character strings that you can input directly using a UTF-compliant text editor, Information Builders recommends that you use the STRREP string replacement function.

The following translates all of the euro signs in a 40-character UTF-8 field to pound sterling signs (£ = C2A3 or 49827):

```
NEWFLD/A40 = CTRAN(40, OLDFLD, EUROVAL, 49827, 'A40');
```

Sort Order Under Unicode

Sort order is based on the binary values assigned to the characters. When FOCUS is configured for Unicode, the sort order is based on the Unicode encoding standard. If ascending values of the codes correspond to the alphabetical order of the letters in the language being used, a report can be sorted in alphabetical order. This is entirely dependent on the encoding standard and its mapping of codes to letters. In many, but not all cases, the encoding standard assigns codes in the alphabetical order of each language.

For example, Ukrainian added a new letter (Cyrillic capital letter ghe with upturn) to its alphabet after the UTF-8 coding specification had already been set. This letter was not assigned a code that sorts it alphabetically, either in Unicode or in code page 1251 (used for Ukrainian). It sorts differently and incorrectly using either encoding scheme.

- With code page 1251, this letter sorts as the first letter on the report output.
- With UTF-8, this letter sorts as the last letter on the report output.

To determine whether a language sorts alphabetically, you can examine the hexadecimal codes assigned to its letters on the code page you are using and check whether ascending hexadecimal codes match the alphabetical order.

Added Unicode Support for Master Files, Data Files, and Application Directory Names

Support for UTF-8 file names is now available for Master Files, Data Files, and Application Directory names.

Unicode PDF Output

Reporting from a Unicode data source with PDF output format is available when using the following two fonts which support Unicode characters:

- Lucida Sans Unicode, which is used to display single-byte characters only. This font is available on Windows version 2000 and higher.
- Arial Unicode MS, which is used to display both single-byte and double-byte characters. This font can be installed as an option from the Microsoft Office CD, version 2000 and higher.

The Lucida Sans Unicode font is the default font if FOCUS is configured for UTF-8 (code page 65001) or UTF-EBCDIC (code page 65002).

If FOCUS is configured for Unicode, and you want to use the Arial Unicode MS font instead, you must specify the Arial Unicode MS font in the StyleSheet. Alternatively, if the FOCUS PDF font mapping file (for example, EDAHOME NLS pdf.fmp) has DEFAULT-FONT=YES specified for Arial Unicode MS, then it becomes the default font. For example:

```
font=Arial Unicode MS, style=normal, metricsfile=PDARUM AFM *,
DEFAULT-FONT=YES, $
```

For more information about how to use the PDF font mapping file, see the *Creating Reports* manual.

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, FOCUS, iWay, Omni-Gen, Omni-HealthData, and WebFOCUS are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (https://www.tibco.com/patents) for details.

Copyright © 2021. TIBCO Software Inc. All Rights Reserved.

Index

-CRTFORM command *29*

-INCLUDE command *244*

-REMOTE BEGIN command *242*, *243*

-REMOTE END command *242*, *243*

-WINDOW command *27*

? command *68*, *251*

? FILEDEF command *118*

? MVS DDNAME command *195*

? MVS DSNAME command *199*

? REMOTE command *245*, *246*

? TSO DDNAME command *154*, *195*

? TSO DSNAME command *199*

?FF command *96*

& line command *68*

&FOCCPU *178*

&FOCZIIPCPU *178*

&FOCZIIPONCP *178*

A

Access Files *120*, *248*

ACCESSDSN field *270*

accessing data *26*

Accessing FOCUS *109*

activating FOCLOG *260*

activating windows *89*, *98*

ACTIVE command *97*, *98*

ADABAS data sources *160*

ADD command *42*

adding lines to files *42*, *43*

ALIAS attribute *249*

ALLOCATE command *140*

- multiple volumes *153*
- UCOUNT *153*
- USERLIB *158*

allocating files *200*

- DYNAM command *203*, *213*
- FMU files *159*
- FOCUS data sources *148*
- multiple units *153*
- multiple volumes *148*
- reviewing attributes *195*
- TRF files *159*
- TSO ALLOCATE command *140*, *179*

allocating FOCLOG log file *258*

allocating WebFOCUS files *113*

application files *113*, *114*, *119*, *143*

- Access Files *120*
- data sources *121*
- external indexes *121*
- procedures *120*
- profiles *111*
- WebFOCUS StyleSheets *122*

applications *25*

AUTOSCROLL parameter *97*, *99*

B

BACKWARD command *51*

BASEDATE field *267*
BASEIO field *268*
batch environment *142*, *148*, *179*
    allocating multi-volume data sources *152*
BATCH field *266*
Beginning a session *110*, *111*
block sizes *148*
    system-determined *163*
borders *100*
BOTTOM command *52*

C

CA-DATACOM/DB data sources *160*
CA-IDMS/DB data sources *160*
CASE command *69*, *70*
case translation functions *322*
CASETBL file *300*, *301*
CDEL command *47*, *49*
CHANGE command *53*, *54*
character semantics and functions *321*, *322*
CINS command *42*, *44*
CLEAR command *97*, *98*
clearing a logical name *119*
clearing windows *98*
client/server architecture *235*
CLIST for FOCLOG reporting *273*
CLIST procedure *182*, *187*
CLOSE command *97*, *100*
closing data sets *222*
CMDCPU field *268*
CMDDURATION field *268*
CMDEXCP field *269*
CMDS RECTYPE *268*
CMDSTART field *268*
COBOL-to-FOCUS Translator *30*
code page definition files *302*, *307*
command environments *16*
COMMAND field *268*
command line commands *78*
command lines *38*, *41*
Command segment *268*
Command window *88*, *90*
commands *90*
    -INCLUDE *244*
    -REMOTE BEGIN *242*
    -REMOTE END *242*
    ? *251*
    ? REMOTE *245*
    ADD *42*
    CASE *69*, *70*
    CDEL *47*
    CHANGE *54*
    CINS *44*
    command line *78*
    CURLINE *44*, *45*
    DELETE *47*
    displaying *68*
    DUPLICAT *58*
    entering *90*
    EX *251*

commands *90*
FFILE *70*, *72*
FILE *70*, *71*
GET *65*
INPUT *44*, *47*
JOIN *21*, *59*
LOCATE *53*
MOVE *58*
OVERLAY *46*
PPUT *63*
PPUTD *64*
prefix area commands *76*
PUT *63*
PUTD *64*
QQUIT *70*, *71*
QUIT *70*, *71*
recalling *105*
RECOVER *47*
REMOTE *246*
REMOTE DESTINATION *239*
REMOTE EX *241*
REMOTE FIN *245*
REMOTE PASSWORD *239*
REMOTE USERID *239*
repeating *68*
REPLACE *46*
SAVE *70*, *72*
scrolling *50*
SPLIT *59*
SSAVE *70*, *72*
commands *90*
truncating *41*
window *97*
compiled window files *158*
compressing data sets *230*
concatenation files *190*
DYNAM command *220*
configuration files *238*
configuration options for FOCLOG *256*
contents in the log file *265*
CONTINUE parameter *97*, *99*
COPY command *55*, *56*
COPY subcommand *223*
copying data sets *223*
copying text *55*, *56*, *58*
CPCODEPG file *294*
CPCPALL file *294*
CPUID field *266*
CPXCPTBL file *294*, *307*
CREATE command *148*
creating a detailed log file *261*
creating files *42*
creating files in TED (text editor) *39*, *40*
CRTFORM command *29*, *40*
CURLINE command *44*, *45*
current language settings *186*
current line *37*
moving *44*
cursors *38*
moving *89*

cursors *38*
positioning in TED (text editor) *38*
customizing editing features *75*
customizing screens *100*

D

data access *26*
data set for logging *255*
data set size *200*
data sets
closing *222*
compressing *230*
copying *223*
deleting *227*
locking *203*
data sources *18*, *26*, *30*, *121*
maintaining *29*
security *26*
joining *21*, *251*
maintaining *28*
UNIX *121*
data transmission *99*
Dataset segment *270*
DB2 tables *160*
DBA attributes
HOLD files *161*
PCHOLD files *161*
DDEXCP field *270*
DDNAME field *270*
ddname queries *196*
ddnames *116*
displaying *118*
decoding, comparison and editing functions *322*
DECRYPT command *156*
default log file *261*
defining file locations *140*
defining files *113*, *116*
defining WebFOCUS files *113*
DELETE command *47*, *49*
DELETE subcommand *227*
deleting text *47*, *49*
DETAIL
specifying detailed log file *261*
detailed log file *261*
developing applications *16*, *25*
DEVFLAG field *270*
diagram of FOCLOG Master File *271*
Dialogue Manager *22*, *26*, *158*
-WINDOW command *27*
displaying error messages *96*
displaying field formats *96*
displaying fields *96*
displaying help *105*
displaying input *93*
displaying output *93*, *99*, *100*
displaying previous commands *68*
displaying reports *95*
distributed execution *236*, *246*, *247*, *249–252*
configuration files for *238*
DNS (Domain Name System) *238*

Domain Name System (DNS) *238*
DOWN command *52*
DSNAME field *270*
DSNORD field *270*
DSNS RECTYPE *270*
DSNTEMP field *270*
DUPLICAT command *55*, *58*
duplicating lines *56*
DYNAM commands *200*
    ALLOCATE *203*, *213*
    allocating multiple volumes *153*
    CLOSE *222*
    COMPRESS *230*
    CONCAT *220*
    COPY *223*
    COPYDD *226*
    DELETE *227*
    FREE *221*
    output printing *205*
    RENAME *228*
    SUBMIT *229*
    syntax *200*, *231*
    UCOUNT *153*
    utility menu *203*
dynamically defining files *115*

E

EDAMAIL *128*
EDIT environment *39*, *41*
editing features *75*
editing files *39*
    multiple *60*
editing procedures *73*
editing text *53*, *54*
ENCRYPT command *156*
ending a session *245*
    TED *70*
entering commands *90*
entry fields *18*
Environmental variables *112*
error messages *96*, *245*
    controlling length *101*
    displaying *96*
Error window *88*, *90*, *96*
ERRORS files
    FOCUS configuration parameters *139*, *142*
ERRORS parameter *97*, *101*
estimating data set sizes *200*
estimating log file size *259*
EX command *251*
example
    FOCLOG validation *264*
Exiting a sessio *111*
Exiting a session *110*
external data sources *160*
external index *155*
external indexes for FOCUS data sources
    UNIX *121*
extract files *113*, *114*, *163*
    DBA security *161*

extract files *113*, *114*, *163*
HOLD *122*
HOLDACC *165*
HOLDMAST *124*, *164*
including comments *161*
SAVB *123*
SAVE *123*
temporary Master Files *124*
UNIX *122*
EXTSORT field *269*

F

FFILE command *70*, *72*
FIDEL environment *29*, *183*
FIDEL screens *40*
field formats *96*
Field window *88*
fields in the log file *265*
file attributes *116*
FILE command *70*, *71*
file for logging *255*
FILE_TYPE field *270*
FILEDEF command *116*, *117*
clear *118*
clearing assignments *119*
displaying assignments *118*
files *21*, *40*
adding lines *42*
adding lines to *42*, *43*
allocating in z/OS *140*, *179*, *203*
files *21*, *40*
allocating under USS *213*
closing *222*
compressing *230*
concatenating with DYNAM command *220*
copying *223*
creating *40*, *42*
creating with INPUT *39*
deleting *227*
deleting lines *49*
editing *39*
editing with TYPE *38*
external *160*
extract *163*
freeing data sets *221*
joining *21*
multiple *60*
naming *138*
renaming *228*
required *142*
scrolling in *50*, *51*
submitting *186*, *229*
transferring text between *62*, *64*, *65*
window *158*
FIN command *180*
Financial Modeling Language (FML) *22*, *125*, *165*
FOCSML files and *125*
Fixed files *321*
Fixed files configuration for Unicode *321*
FLVALPOP procedure *262*

FLVALRPT procedure *262*
FML (Financial Modeling Language) *22*, *125*, *138*, *165*
    FOCSML files and *125*
FMU files *159*
FNAME field *270*
FOC$HOLD files *122*
FOCALLOC parameter *148*
FOCEXEC field *268*, *269*
FOCEXECDSN field *269*
FOCEXECs *16*, *78*, *83*
FOCEXECs for FOCLOG
    FLVALPOP *262*
    FLVALRPT *262*
FOCLOG activation *260*
FOCLOG configuration options *256*
FOCLOG description *253*
FOCLOG differences *253*
FOCLOG implementation *254*, *257*
FOCLOG installation *257*
FOCLOG log data set *255*
FOCLOG log file size
    estimating *259*
FOCLOG log file
    allocating *258*
FOCLOG Master File *266–268*
    Dataset segment *270*
    Master segment *270*
    structure diagram *271*
FOCLOG overview *253*
FOCLOG procedures
    FLVALPOP *262*
    FLVALRPT *262*
FOCLOG reporting *265*, *273*
FOCLOG reporting CLIST *273*
FOCLOG validation *262*
FOCLOG
    validating *264*
FOCPOST files *125*, *167*
FOCREL field *266*
FOCSML files *125*, *167*
    FML and *125*
FOCSORT files *125*, *166*
    allocating multiple units *153*
    allocating multiple volumes *148*
FOCSORT maximum size *166*
FOCUS *15*, *16*
    in the Linux environment *125*
    in the UNIX environment *125*
    sessions *17*
fonts *324*
    Unicode in PDFs *324*
format conversion functions *322*
FORWARD command *51*
freeing data sets *221*
FSCAN facility *29*
function keys *41*, *42*, *72*, *76*
    defining *75*, *94*, *101*
    PF10 *68*, *69*
    PF11 *68*, *69*

function keys *41*, *42*, *72*, *76*
PF19 *50*, *51*
PF2 *43*
PF20 *50*, *51*
PF23 *68*
PF3 *70*, *71*
PF5 *68*
PF6 *68*
PF7 *50*, *51*
PF8 *50*, *51*
functions *25*
FUSELIB (FOCUS User-Written Subroutine Library) *158*
FX command *192*

G

GDG DYNAM support for relative number *205*
GET command *62*, *65*
global profiles *111*
group profiles *111*

H

HELP command *97*, *105*
HELP files *72*
Help window *88*, *90*, *94*, *105*
History window *88*, *90*, *93*
HLRECL field *269*
HOLD files *163*
UNIX *122*
HOLDACC files *165*
HOLDMAST files *163*, *165*
UNIX *124*
HOLDSTAT files *139*, *161*
host names *238*
HOST parameter *238*
Hot Screen facility *93*

I

IBIREG field *267*
IEDIT command *83*, *84*
IMMEDTYPE parameter *97*, *99*, *100*
implementing FOCLOG *257*
activating *260*
allocating log file *258*
estimating log file size *259*
overview *257*
validating *262*
IMS data sources *160*
INPUT command *42*, *44*, *47*
INPUT environment *39*
inserting text *47*
inserting text in files *45*
installing FOCLOG *257*
activating *260*
allocating log file *258*
estimating log file size *259*
validating *262*
installing IEDIT *84*
interoperability *235*
interrupt commands *88*, *190*

Interrupting a session *111*
ISPF EDIT command *84*
ISPF edit screens *193*
ISPF statistics *185*
iWay data adapters *246*
iWay middleware *235*

J

JCL command *140*, *231*
JOBID field *267*
JOBNAME field *267*
JOIN command *21*, *59*, *60*
joining data sources *251*
joining files *21*
joining lines *59*, *60*
justification functions *322*

K

known code page files *307*
KX command *190*

L

LEFT command *68*, *69*
LEFTP command *68*, *69*
length and position functions *322*
LET files *165*
libraries
    FUSELIB *158*
    STEPLIB *181*
    USERLIB *158*
limits
    FOCSORT file size *166*
    FOCUS file size *147*
line numbers *66*
LINENUM field *268*
lines *42*
    adding to files *42*, *43*
    duplicating *56*
    joining *59*, *60*
    splitting *59*, *60*
Linux
    concatenation symbol *127*
    defining files *113*
    environment for FOCUS *125*
    extract files *113*
    FOCUS bacground process *128*
    function keys *127*
    international 8-bit character set *127*
    remapping keys *126*
    teminal support *126*
LOCATE command *53*
locating text *53*
location transparency *249*
locking data sets *203*
log file *255*, *265*
    allocating *258*
    Command segment *268*
    creating default log *261*
    creating session summary log *261*
    Dataset segment *270*

log file *255*, *265*
Master segment *270*
Session segment *266–268*
LOG files *165*
logging on to servers *239*, *250*
logical names *116*, *119*
FILEDEF command and *116*, *117*
logical records *20*
LOWERCAS command *70*
lowercase text *69*
LPARNAM field *267*

M

Maintain facility *27*
maintaining data sources *28*
MASS RECTYPE *270*
Master File fields *266–268*
Master File for FOCLOG
structure diagram *271*
Master File
fields for FOCLOG *270*
Master Files *119*, *248*
HOLDMAST files *124*
in z/OS *144*
Master segment for FOCLOG *270*
MASTERDSN field *270*
menus *27*
MICROSEC field *266*
Microsoft SQL Server configuration for Unicode *317*
middleware *235*
Millennium data sources *160*
MODEL 204 data sources *160*
MODEL_ID field *267*
MODEL_NUM field *267*
MODIFY facility *27*
Monitor for zIIP processing *177*
monitoring resource usage *29*
monocasing *300*
customizing for z/OS *300*, *301*
tables *301*
MOVE command *55*, *58*, *100*
moving text *55*, *58*, *59*
moving the screen display *68*, *69*
MSO field *266*
multi-image FOCSORT *166*
multi-volume support *148*
multiple files *60*
multiple units for FOCUS and FOCSORT *153*
multiple volumes for FOCUS and FOCSORT *148*
MVS prefix *187*

N

naming conventions *138*
National Language Support (NLS) *186*, *291*
customizing monocasing tables *300*, *301*
NEXT command *52*, *97*, *98*
NLS (National Language Support) *186*, *291*
adding code pages *296*
code page definition files *307*

NLS (National Language Support) *186*, *291*
configuration files *303*
configuring language settings *299*
CPXCPTBL file *307*
customizing monocasing tables *300*, *301*
customizing sort sequences *302*
default code pages *295*
NLSCFG configuration file *305*
PC3270 configuration *299*
steps *296*, *300*
TSGU *308*
verifying configuration *299*
NLS configuration files *291*, *294*
NLS sort tables *302*
NLSCFG file *304*
nlscfg.err file *305*
non-procedural languages *16*
NOPROF parameter *180*
NOREMOTEECHO parameter *237*
NUM command *66*, *67*

O

OFFLINE field *269*
OFFLINE files
UNIX *125*
offline printing
DYNAM ALLOCATE operands *205*
output files *141*
offload processing to zIIP engine *168*
OPEN command *97*, *100*
OPSYS field *266*
OPSYSREL field *266*
Oracle character semantics *315*
Oracle configuration for Unicode *315*
Oracle tables *160*
OUTFLAG field *269*
output *93*
displaying *93*, *99*, *100*
OUTVOLUME field *269*
OVERLAY command *46*
overlaying text *46*
overview of FOCLOG *253*, *254*

P

page formulas *200*
PAINT environment *40*
parameter settings *251*
passthru *252*
passwords *239*
setting *239*, *250*
PATH *109*
PC3270 code page setting *299*
PDF report output *324*
Unicode fonts *324*
PF function keys *76*
PF key functions *72*
PF key settings *94*
PF10 function key *68*, *69*
PF11 function key *68*, *69*
PF12 function key *98*

PF19 function key *50*, *51*
PF2 function key *43*
PF20 function key *50*, *51*
PF23 function key *68*
PF3 function key *70*, *71*
PF5 function key *68*
PF6 function key *68*, *105*
PF7 function key *50*, *51*
PF8 function key *50*, *51*
POOLED field *269*
populating the log *262*
POST files *165*
PPUT command *62*, *63*
PPUTD command *62*, *64*
prefix area commands *39*, *43*, *44*, *47*, *48*, *56–60*, *65*, *76*
printing
    using DYNAM *205*
procedural languages *16*
procedures *16*, *26*, *145*, *146*
    editing *73*
    running *241–243*
processing
    offloading to zIIP engine *168*
PROCESSOR_ID field *268*
PROFILE procedures *180*
profiles *75*, *146*
    TED *186*
PUT command *62*, *63*
PUTD command *62*, *64*

Q

QQUIT command *70*, *71*
query commands *245*, *251*
    ? FILEDEF *118*
    ? MVS DDNAME *195*
    ? MVS DSNAME *199*
    ?FF *96*
    REMOTE *245*, *246*
QUIT command *70*, *71*

R

READS field *268*
REBUILD command *155*
    work files *167*
RECALL command *97*, *105*
recalling commands *105*
RECORDS field *268*
RECOVER command *47*, *49*
recovering text *47*, *49*
RECTYPE *266*
RECTYPE CMDS *268*
RECTYPE DSNS *270*
RECTYPE field *268*
RECTYPE MASS *270*
RECTYPE SESS *266*
referencing files
    z/OS *138*
regular expressions *31*
remote data access *235*
REMOTE DESTINATION command *239*, *240*

REMOTE EX command *241*, *242*
remote execution *236*, *239*, *241–243*
    configuration files for *238*
REMOTE FIN command *245*
REMOTE PASSWORD command *239*
remote procedure calls (RPCs) *236*, *243*, *244*, *251*
remote servers *245*
    ending a session *245*
REMOTE USERID command *239*
renaming data sets *228*
repeating previous commands *68*
REPLACE command *46*
replacing text *45*, *46*
Report Writer *22*
reporting *273*
reporting from the FOCLOG log *262*
reports
    displaying *95*
requests *90*
    entering *90*
requirements for FOCUS *142*
Resource Governor *29*
resources *29*
    monitoring usage *29*
RESTRICT command *157*
RIGHT command *68*
RIGHTP command *68*, *69*
ROUTE command *97*, *106*
RPCs (remote procedure calls) *236*, *243*, *244*, *251*
RT command *192*
RUN command *73*

S

sample FOCLOG validation session *264*
SAP BW configuration for Unicode *320*
SAP R/3-ECC configuration for Unicode *320*
SAVB files *164*
    UNIX *123*
SAVE command *70*, *72*
SAVE files *164*
    UNIX *123*
SBORDER parameter *100*
SCALE command *67*
scales *66*
SCAN line editor *29*
screen forms *29*
Screen Painter *40*
screens *40*
    creating *40*
    customizing *100*
    moving the display *68*, *69*
    splitting *61*, *62*
SCROLL command *97*, *106*
scrolling commands *50*
    BACKWARD *51*
    BOTTOM *52*
    DOWN *52*

scrolling commands *50*
FORWARD *51*
NEXT *52*
TOP *52*
UP *52*
scrolling in files *50*
scrolling the Output window *99*
scrolling window contents *106*
security *25*, *26*
DBA attributes in extract files *161*
segments *18*, *266–268*
Command for FOCLOG *268*
Dataset for FOCLOG *270*
sequential data sources *121*
UNIX *121*
sequential files *146*
SERIAL_NUM field *267*
SERVER parameter *246*, *250*
servers *239*
locating *246*, *247*
logging on *239*, *250*
SESCPU field *266*
SESDURATION field *266*
SESEXCP field *266*
SESS RECTYPE *266–268*
session parameters *245*
displaying *245*, *246*
Session segment *266–268*
Session segment for FOCLOG *266–268*
session summary log file *261*
SESSION
specifying session summary log file *261*
sessions *70*, *93*
ending *70*
viewing history of *93*
SESSTART field *266*
SET command *101*
SET parameters *97*, *99*
AUTOSCROLL *99*
CONTINUE *99*
FOCALLOC *148*
HOLDSTAT *161*
IMMEDTYPE *100*
NOREMOTEECHO *237*
SBORDER *100*
SERVER *246*, *250*
SIZE *101*
USAGEFORMAT *237*
USER *250*
ZIIP *173*
SHADOW parameter *200*
Shell scripts
focus *110*
focus command parameters *112*
SIMMAXZIIP *173*
sink machine *133*
SINKID field *270*
SITECODE field *266*
size of log file *259*
SIZE parameter *97*, *101*

SmartMode option *29*
SMTP server configuration *129*
sort order with Unicode *324*
sorting tables *302*
SORTIO field *268*
SORTOUT files *155*, *167*
source code *237*
specifying default log file *261*
SPH command *61*
SPLIT command *59*, *60*
SPLITH command *61*
splitting lines *60*
splitting screens *61*, *62*
SPLITV command *62*
SPV command *62*
SQL commands *237*, *251*
    ? *251*
    EX *251*
    USAGEFORMAT *237*
SQL passthru *252*
SSAVE command *70*, *72*
STACK files *166*
STARTDATE field *267*
STARTDT field *266*
STARTHOUR field *267*
Starting a session *110*, *111*
STARTMONTH field *267*
STARTQTR field *267*
STARTWEEK field *267*
STARTYYMTR field *267*
statistics *192*
    ISPF *185*
STEPLIB libraries *181*
stored procedures *26*, *243*, *251*
    remote execution *243*, *244*
string manipulation and extraction functions *322*
string replacement functions *322*
structure diagram of FOCLOG Master File *271*
submitting jobs *186*
    DYNAM command *229*
    TSO SUBMIT command *179*
subroutines
    FUSELIB *158*
SUFF field *270*
SUFFIX attribute *246*
    EDA *246*, *247*
suppressing source code *237*
SUPRA data source *160*
Sybase ASE Adapter *317*
    Unicode support *317*
SYSTEM 2000 data sources *160*
system editor *83*
system messages *245*
system requirements for zIIP enablement *170*

T

Table window *88*, *90*, *95*
TableTalk sessions *138*
    application files *160*
    work files *168*

TABLTALK files *168*
TED (text editor) *22*, *35*, *37*, *184*, *185*
  command line *38*
  current line *37*
  EDIT environment *39*
  environments *75*
  positioning cursors in *38*
  profiles *186*
  screen layout *37*
  TYPE environment *38*
temporary Master Files *124*
temporary storage *62*
Teradata data sources *160*
Terminal Operator Environment (TOE) *17*, *87*, *88*
terminal support *110*
text *45*
  copying *55*–*58*
  deleting *47*, *49*
  editing *53*, *54*
  inserting *45*, *47*
  locating *53*
  moving *55*, *58*, *59*
  overlaying *46*
  recovering *47*, *49*
  replacing *45*, *46*
  specifying case *69*
  transferring between files *62*–*65*
TOE (Terminal Operator Environment) *17*, *87*, *88*
TOE windows *88*
  Command *88*, *90*
TOE windows *88*
  commands *88*, *97*
  Error *88*, *90*, *96*
  Field *88*
  Help *88*, *90*, *94*, *105*
  History *88*, *90*, *93*
  Output *88*, *90*, *93*, *99*
  Table *88*, *90*, *95*
TOP command *52*
TOTAL files *160*
transcoding *308*
  generating tables *308*
transferring text between files *62*–*65*
TRF files *159*
truncating commands *41*
TSGU (Transcoding Services Generation Utility) *308*
  help information *311*
  info-commands *308*, *310*
TSGU info-commands *308*
TTEDIT files *160*
TYPE environment *38*, *41*
TYPE messages *165*

U

UCOUNT command *203*
  allocating multiple units *153*
Unicode *313*
  access non-FOCUS data sources *315*
  Asian characters and *313*

Unicode *313*
data type support for Relational adapters *319*
DB2 data types *319*
European characters and *313*
Fixed files configuration *321*
fonts for PDFs *324*
manipulating characters *321*
Microsoft SQL data types *319*
Microsoft SQL Server configuration *317*
Oracle configuration *315*
Oracle data types *319*
PDF output *324*
reformatting characters *321*
SAP BW configuration *320*
SAP R/3-ECC configuration *320*
sort order and *324*
Sybase ASE Adapter *317*
Sybase ASE data types *319*
unit count *203*
units *152*
allocating FOCUS data sources *148*
allocating multiple units *153*
universal character encoding standard *313*
UNIX directory permissions *110*
UNIX environment for FOCUS *110*
UNIX PATH *109*
UNIX procedures *120*
UNIX shell scripts
focus *110*
focus command parameters *112*
UNIX terminal support *110*
UNIX variables *112*
UNIX
concatenation symbol *127*
data sources *121*
defining files *113*
environment for FOCUS *125*
external indexes *121*
extract files *113*, *122*
FOCUS bacground process *128*
FOCUS operating environment *112*
FOCUS session *110*
function keys *127*
international 8-bit character set *127*
profile processing *111*
profiles *111*
remapping keys *126*
teminal support *126*
WebFOCUS StyleSheets *122*
UP *52*
UPPERCAS command *70*
uppercase text *69*
USAGEFORMAT parameter *237*
user authentication *239*, *250*
USER parameter *250*
user profiles *111*
user-written functions *25*
USERID field *266*
USERLIB libraries *158*
using IEDIT *84*

USS
allocating files *213*
defining files *213*
UTF (Unicode Transformation Format) *313*

V

validating FOCLOG *262*
populating the log *262*
reporting from the log *262*
sample session *264*
variable substitutions *204*
DYNAM ALLOCATE operands *204*
variables *197*
verifying NLS configuration *299*
viewing reports *95*
VOLSER (volume serial number) variable *154*
allocating multi-volume data sources *152*
volume *149*
allocating multiple *148*
identifiers *154*

W

WebFOCUS files *113*
dynamically defining *115*
extract files *122*
types *113*, *114*
WebFOCUS StyleSheets
UNIX *122*
WINDOW command *88*, *97*
window documentation files *158*
WINDOW HELP command *94*
Window Painter *27*, *158*
allocating FMU files *159*
allocating TRF files *159*
WINDOW SET ERRORS command *96*
windows *27*
activating *89*, *98*
clearing *98*
creating *27*
displaying *100*
enlarging *105*
hiding *100*
moving *100*
scrolling *106*
sizing *101*
transferring contents *106*
Winform Painter *27*
work areas in Terminal Operator Environment *88*
work files *113*, *114*
FOCPOST *125*
FOCSML *125*
FOCSORT *125*
OFFLINE *125*

X

XFOCUS data sources *121*, *148*

Z

z/OS *84*, *137*, *138*
allocating files *140*, *152*, *179*, *203*

z/OS *84*, *137*, *138*
- batch *144*, *148*, *179*
- changing code page settings *293*
- commands *187*, *188*
- comparing command syntax *231*
- configuring for NLS *291*
- data set sizes *200*
- defining files *203*
- FOCUS *179*
- identifying page settings *295*
- interrupting FOCUS sessions *190*
- LOGON procedures *181*
- prefix area *186*
- PROFILE procedures *180*

z/OS *84*, *137*, *138*
- referencing files *138*
- submitting jobs *229*
- system variables *197*
- updating ISPF statistics *185*

zIIP enablement *168*, *177*
- data sources *178*
- requirements for *169*
- system requirements *170*
- types of processing offloaded *177*

zIIP Monitor *177*

ZIIP parameter *173*

ZOOM command *97*, *105*