Information Builders

FOCUS for Mainframe

Overview and Operating Environments

Version 7.6

DN1001055.0108

Contents

Preface

This documentation describes how to use FOCUS Version 7.6 in the z/VM (CMS) and z/OS environments. It is intended for all FOCUS users. This manual is part of the FOCUS documentation set.

References to z/OS apply to all supported versions of the OS/390, z/OS, and MVS operating environments. References to z/VM apply to all supported versions of the VM/ESA and z/VM operating environments.

The documentation set consists of the following components:

- *Creating Reports* describes FOCUS Reporting environments and features.
- *Describing Data* explains how to create the metadata for the data sources that your FOCUS procedures will access.
- *Developing Applications* describes FOCUS Application Development tools and environments.
- *Maintaining Databases* describes FOCUS data management facilities and environments.
- *Using Functions* describes internal and user-written functions.
- *Overview and Operating Environments* contains an introduction to FOCUS and FOCUS tools and describes how to use FOCUS in the z/VM CMS and z/OS environments.

The users' documentation for FOCUS Version 7.6 is organized to provide you with a useful, comprehensive guide to FOCUS.

Chapters need not be read in the order in which they appear. Though FOCUS facilities and concepts are related, each chapter fully covers its respective topic. To enhance your understanding of a given topic, references to related topics throughout the documentation set are provided. The following pages detail documentation organization and conventions.

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Introduction to FOCUS	Provides an overview of FOCUS.
2	Editing Files With TED	Describes the FOCUS text editor, TED, and shows how to edit text and data sources.
3	Invoking Your System Editor With IEDIT	Describes the IEDIT command, which opens the system editor from within FOCUS. This feature enables you to edit and execute variable length files and those with record lengths longer than 160 bytes, which TED does not support.
4	Terminal Operator Environment	Describes the optional windowed environment for running FOCUS.
5	CMS Guide to Operations	Is a guide to operations for FOCUS users who run under CMS. It explains how to define files, describes all FOCUS application, extract, and work files, and discusses how FOCUS facilities interact with the operating system.
6	z/OS Guide to Operations	Is a guide to operations for FOCUS users who run under z/OS. It describes how to allocate files, describes all FOCUS application, extract, and work files, and discusses how FOCUS facilities interact with the operating system.
7	Recording and Replaying a FOCUS Session	Describes FOCREPLAY, a facility for recording an interactive session and playing it back in batch mode.
8	Using FOCUS as a Client to a Reporting Server	Describes FOCUS client/server computing and elements of Information Builders' Middleware Technology.
9	Logging FOCUS Usage: FOCLOG	Describes FOCLOG, a tool for recording and analyzing the use of FOCUS for your entire site.
10	Storing Terminal Lines in Memory: The Session Monitor	Describes Session Monitor, a facility for reviewing and saving the input and output generated during online FOCUS sessions.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must enter exactly as shown.
`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
`.` `.` `.`	Indicates that there are (or could be) intervening or additional commands.

Related Publications

To view a current listing of our publications and to place an order, visit our World Wide Web site, http://www.informationbuilders.com. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about FOCUS?

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

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

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

Information You Should Have

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

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

- ? SET/? SET GRAPH
- ? USE
- ? TSO DDNAME OR CMS FILEDEF

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

User Feedback

In an effort to produce effective documentation, the Documentation Services staff welcomes your opinions regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Documentation Feedback form on our Web site, http://www.informationbuilders.com.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

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

1 Introduction to FOCUS

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

Topics:

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

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 Chapter 4, *Terminal Operator Environment*.

FOCUS Concepts

In this section:

Combining Data From Several Data Sources

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.

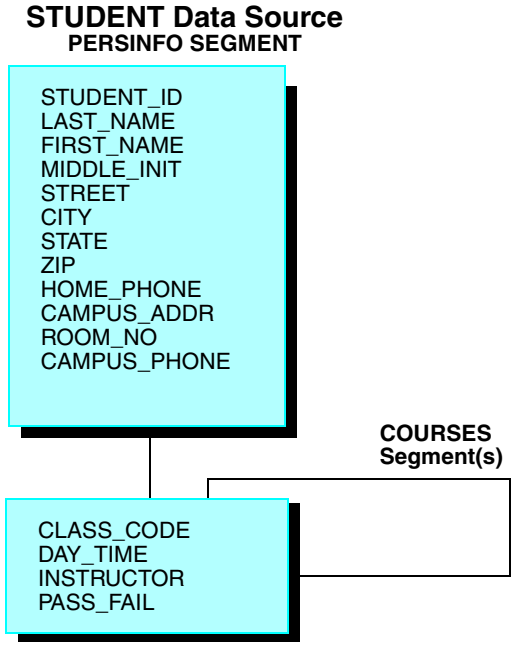
In simple applications, data sources may consist of a single segment, but generally more than one segment is needed. Consider an additional screen that captures information about each course a student selects:

```
                          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).

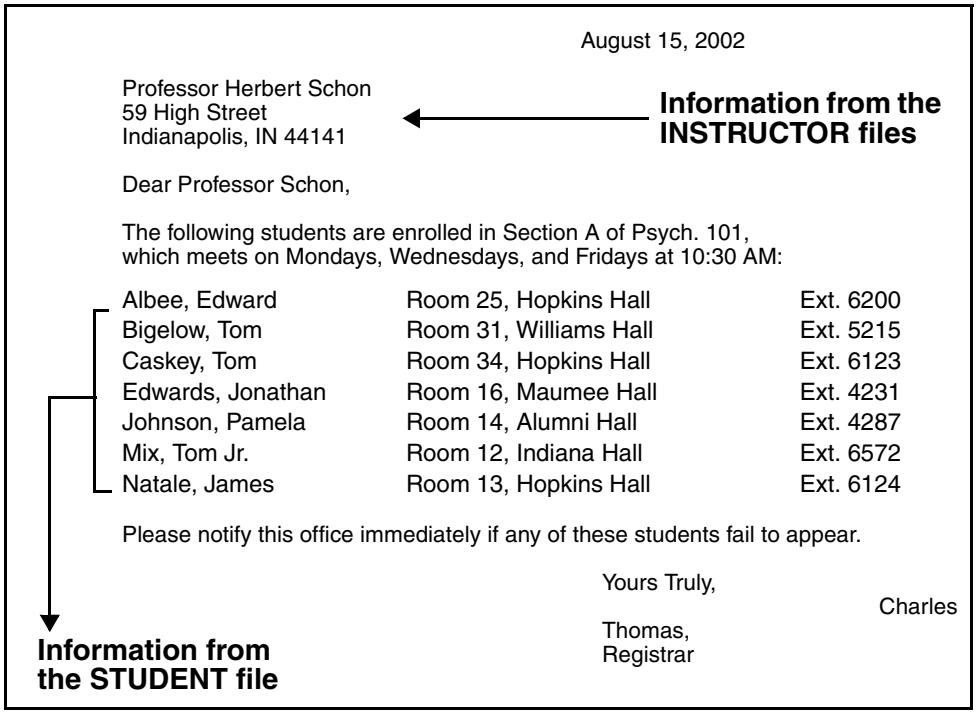
The data (the actual pieces of information described by the entries in the Master File, such as a student's name and Student Identification Number) exist in logical records in the data source. For example, all of the information about a student in the STUDENT data source is in a single logical record. 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

In this section:

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 Chapter 2, *Editing Files With TED* and Chapter 3, *Invoking Your System Editor With IEDIT*.
- 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.*
- ICU and CA-TELLAGRAF Interface manuals.

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.

If you have the FOCUS CA-TELLAGRAF Interface (described in a separate manual), you can generate graphic output correctly formatted for use by CA-TELLAGRAF, a publication-quality graphics package from Computer Associates.

If you have the Interactive Chart Utility (ICU) Interface (described in a separate manual), you may use ICU to format graphs in conjunction with GRAPH syntax.

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 Chapter 2, *Editing Files With TED*. The IEDIT facility is described in Chapter 3, *Invoking Your System Editor With IEDIT*.

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
  ```

  FOCUS invokes the editor and displays your request on the edit screen with the cursor on the error line.
- ❏ 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.
- ❏ Format FOCUS request output for use by CA-TELLAGRAF or ICU.
- ❏ 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

In this section:

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, DB2 for VM, Oracle, Teradata, CA-IDMS/DB, IMS/DB, MODEL 204, and Millennium.
- 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 SmartMode's protection of the site's most important resources. Reports and graphs on resource usage patterns show how data sources are used, who uses them, and how resource usage is distributed.

SmartMode is available for z/OS and CMS. For complete documentation and installation instructions, see the *SmartMode for FOCUS Installation and Operations Manual* for your operating environment.

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.

You may run a FOCUS session by selecting options from menus in window-based shells:

- The FOCUS Menu screen provides a convenient way to access FOCUS environments using windows.
- The FOCUS Toolkit also provides easy access to FOCUS environments plus tools to perform decision support, data maintenance, and administrative tasks.

For information about the FOCUS Menu and the FOCUS Toolkit, see the Guide to Operations chapter for your site's operating system.

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.*

2 Editing Files With TED

TED is a general-purpose text editor you can use to create or edit files while in FOCUS. It is a full-screen editor that enables you to insert, delete, and replace characters anywhere on the screen.

These topics describe the four TED editing environments and their use for creating and modifying Master Files, report 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 XEDIT and ISPF editors that run under CMS and TSO. While those editors 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.
- Immediate execution of FOCEXECs and the facility to recall the line of error in a FOCEXEC.
- The same editor in every environment for multi-environment FOCUS users.

Note: TED supports editing of files with LRECL up to 160. To edit a file whose record length is longer than 160, use the IEDIT facility. For more information, see Chapter 3, *Invoking Your System Editor With IEDIT*.

Topics:

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. The naming conventions of the file vary with the operating system you are using.

For example, to edit a FOCEXEC in CMS, enter:

```
TED filename [FOCEXEC]
```

See Chapter 5, *CMS Guide to Operations*, for more information.

In z/OS, enter:

```
TED FOCEXEC(member)
```

or

```
TED name
```

See Chapter 6, *z/OS Guide to Operations*, for more information.

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

In this section:

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 file name in CMS or the data set name or ddname in z/OS.
- ❏ The file type and file mode (for CMS users).
- ❏ The size (the number of lines in the file).
- ❏ The line number of the current line.

The first screen in CMS is:

```
EXAMPLE DATA             A1            SIZE=0            LINE=0

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




====>
                                                            TED
```

The first screen in z/OS is:

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

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



====>
                                                          TED
```

Note: Screens used in these topics are from a CMS operating system; the only difference between a CMS and z/OS screen display in TED is the file name.

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 45.

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.
Æ	Moves the cursor up.
¦	Moves the cursor to the right.
Ø	Moves the cursor to the left.
Tab	Moves the cursor to the next line in TYPE or to the next Tab stop in EDIT.
Back Tab	Moves the cursor to the previous line.
Home	Moves the cursor to the top of the screen.
Return	Moves the cursor to the next line.

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 80). TED enables you to create lines up to 160 characters long in CMS (159 in z/OS). 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 43.

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:

```
EXAMPLE   DATA            A1      SIZE=0        LINE=0

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



====>
                                                  EDITING MODE
```

EDIT is fully explained in *TYPE and EDIT Functions* on page 43.

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 43).

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:

```
EXAMPLE DATA        A1            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 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 * * *
                                                        INPUT-MODE
```

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:

```
EXAMPLE DATA      A1              SIZE=5          LINE=0


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



====>
                                                TYPING MODE
```

TYPE and EDIT Functions

In this section:

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

In this section:

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):

```
EXAMPLE DATA      A1                SIZE=0       LINE=0

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



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

After pressing the Enter key, five lines are added, as shown below.

```
EXAMPLE DATA      A1               SIZE=5        LINE=1

===== * * * 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

In this section:

:n

±n

==/==

CUrline

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:

```
EXAMPLE DATA        A1            SIZE=3          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

In this section:

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:

```
EXAMPLE DATA       A1             SIZE=3         LINE=0

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:

```
EXAMPLE DATA       A1             SIZE=3         LINE=2

THIS WILL BE CHANGED TO THE LINE ABOVE BUT 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

In this section:

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:

```
EXAMPLE DATA        A1     SIZE=3          LINE=1

===== * * * 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:

```
EXAMPLE DATA         A1     SIZE=0            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:

```
EXAMPLE DATA       A1      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.

```
EXAMPLE DATA       A1      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

In this section:

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
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

In this section:

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 72). 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:

```
EXAMPLE DATA         A1    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.

```
EXAMPLE DATA       A1      SIZE=6        LINE=6

OCCURRENCE(S) CHANGED: 4
===== 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

In this section:

The following commands duplicate and move text in a file:

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

```
EXAMPLE MASTER     A1     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:

```
EXAMPLE DATA       A1     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 this section:

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.

```
EXAMPLE DATA       A1      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:

```
EXAMPLE DATA       A1       SIZE=3        LINE=1

===== 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

In this section:

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:

```
EXAMPLE DATA        A1     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
--------------------------------------------------------------------------------
EXAMPLE DATA        A1     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
```

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:

CMS

```
TEd filename
```

TSO

```
TED ddname(member)
```

where:

filename

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

Transferring Text Between Files and Temporary Storage

In this section:

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

Command Line Commands	Prefix Area Commands
PUT	==PP=
PPUT	==PL= ==Pn=
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 name of the file where you want to store the lines of text. In z/OS, the file name is ddname(member name). 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:

```
EXAMPLE DATA      A1       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, ==Pn=

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 name of the file where you want to store the lines of text. 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:

```
NEWFILE DATA       A1      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:

```
NEWFILE DATA       A1      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 name of the file that contains the text. In z/OS, the file name is ddname(member name). 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 64.
3. Retrieve the lines from temporary storage using the GET command.

Displaying a Scale and Line Numbers

In this section:

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:

```
EXAMPLE DATA       A1     SIZE=5          LINE=0

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

```
EXAMPLE DATA       A1      SIZE=4  LINE=0

....+....1....+....2....+....3....+....4....+....5....+....6....+....7
===== * * * TOP OF FILE * * *
===== THIS SCREEN SHOWS A SCALE
=====
=====
=====
===== * * * 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 this section:

In TED, you may create lines of up to 160 characters in CMS (159 in TSO). 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

In this section:

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

In this section:

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:

```
EXAMPLE MASTER     A1     SIZE=7        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 EXAMPLE MASTER
                                                    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:

```
M1-1 FOCEXEC       A1      SIZE=10         LINE=0

* * * TOP OF FILE * * *
MODIFY FILE EMPLOYEE
FREEFORM EMP_ID CURR_SAL
MATCH EMP_ID
 ON NOMATCH REJECT
 ON MATCH UPDATE CURR_SAL
DATA
EMP_ID=071382660, CURR_SAL=21400.50, $
EMP_ID=112847612, CURR_SAL=20350.00, $
EMP_ID=117593129, CURR_SAL=22600.34, $
END
* * * 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.

Note: In CMS you cannot issue the RUN command from TED unless the file type of your file is FOCEXEC. This is not required in z/OS.

In the following example, the FOCEXEC is missing the file name:

```
M1-1 FOCEXEC      A1       SIZE=10        LINE=0

* * * TOP OF FILE * * *
MODIFY FILE
FREEFORM EMP_ID CURR_SAL
MATCH EMP_ID
 ON NOMATCH REJECT
 ON MATCH UPDATE CURR_SAL
DATA
EMP_ID=071382660, CURR_SAL=21400.50, $
EMP_ID=112847612, CURR_SAL=20350.00, $
EMP_ID=117593129, CURR_SAL=22600.34, $
END
* * * END OF FILE * * *

====> RUN
                                                        EDITING MODE
```

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

```
ERROR AT OR NEAR LINE 2 IN PROCEDURE M1-1 FOCEXEC *

(FOC205) DESCRIPTION NOT FOUND FOR FILE NAMED: FREEFORM
BYPASSING TO END OF COMMAND
>
```

If you enter 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 EMP_ID CURR_SAL.

```
M1-1 FOCEXEC       A1      SIZE=10       LINE=2

FREEFORM EMP_ID CURR_SAL
MATCH EMP_ID
 ON NOMATCH REJECT
 ON MATCH UPDATE CURR_SAL
DATA
EMP_ID=071382660, CURR_SAL=21400.50, $
EMP_ID=112847612, CURR_SAL=20350.00, $
EMP_ID=117593129, CURR_SAL=22600.34, $
END

* * * 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.

In CMS, the file name must be:

```
PROFILE TED
```

In TSO, the file name must be

```
FOCEXEC(TEDPROF)
```

where:

`TEDPROF`

Is a member of the PDS allocated to ddname FOCEXEC.

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

```
PFnn command
```

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 causes an error.

Syntax Summary

In this section:

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.
==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.
`CMS command`	Issues CMS command from TED (CMS users only).
`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 filename`	Saves file as *fileid* and ends session (CMS).
`FIle ddname(member)`	Saves file as *member* and ends session (z/OS).

Command	Action
FILEName *newfilename*	Changes the default file name used for FILE and SAVE commands (CMS).
FILEName *ddname* (new member)	Changes the default member used for FILE and SAVE commands (z/OS).
FILEType *newfiletype*	Changes the default file type used for FILE and SAVE commands (CMS only).
FILEMode *newfilemode*	Changes the default file mode used for FILE and SAVE commands (CMS only).
FN *newfilename*	See FILENAME command.
FT *newfiletype*	See FILETYPE command.
FM *newfilemode*	See FILEMODE command.
FOrward *n*	Moves forward *n* pages.
Get *filename*	Gets a file or gets stack if no fileid given (CMS).
Get *ddname*(*member*)	Gets a member or gets stack if ddname and member are not specified (z/OS).
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.

Command	Action
`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.
`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 (z/OS users only).
`TEd filename`	Edits another file from within the current file (CMS).

Command	Action
`TED ddname(member)`	Edits another file (z/OS).
`Top`	Goes to top of file.
`TSO command`	Issues TSO command (z/OS users only).
`TYpe`	Sets mode to data display, no prefix area.
`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*.

3 Invoking Your System Editor With IEDIT

The IEDIT command opens the system editor (XEDIT on CMS or ISPF EDIT on z/OS) from within FOCUS. This feature enables you to edit and execute files with record lengths longer than 160 bytes, which TED does not support.

Topics:

Editing Files With IEDIT

How to:

Edit Files With IEDIT

Reference:

Usage Notes for IEDIT

Example:

Editing a File With IEDIT on z/OS

Editing a File With IEDIT on CMS

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

On CMS:

```
IEDIT [filename [[filetype|FOCEXEC] [filemode|A]]]
[(any_valid_XEDIT_option
```

On z/OS:

```
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.

filetype

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

filemode

Is the file mode of the file to edit. If the file exists on any disk, the mode of this file is used. If no file is found, it a new file is opened on A.

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 on z/OS

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
```

Example: Editing a File With IEDIT on CMS

To edit a Master File (FILETYPE MASTER) named CENTORD:

```
IEDIT CENTORD MASTER
```

To edit a FOCEXEC named LOADORD:

```
IEDIT LOADORD
```

To edit the last FOCEXEC that was run:

```
IEDIT
```

IEDIT Facilities on CMS

When making changes to a FOCEXEC or an EXEC, the RUN command can be issued from the XEDIT command line. This is similar to the RUN command in TED. RUN can be issued for FOCEXECs and EXEC files.

RUN performs the following operations:

- Saves the file.
- Executes a FOCEXEC and passes to it any parameters specified.
- Executes a CMS EXEC with an extended plist.

Installing IEDIT on z/OS

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 ISPTLIB.

- **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.

Installing IEDIT on CMS

Four files are required for using IEDIT:

- **CONFIG IEDIT.** This is the IEDIT profile shipped with Information Builders defaults that executes when IEDIT is invoked. This file should not be modified.
- **PROFILE IEDIT.** This file is for user-defined IEDIT customization. Out of these IEDIT configuration files, this is the only one that users should edit. For example, you can add:

  ```
  SET AUTOSAVE 5
  ```

- **RUN IEDIT.** This is a macro that enables the RUN command, and should not be modified.
- **IEDIT CONFIG.** This is a REXX EXEC that is the interface between FOCUS and IEDIT. This file should not be modified.

A user can customize the look and feel of the XEDIT environment by making a copy of PROFILE IEDIT on the A disk, and making changes as needed. All other files with a filename or filetype of IEDIT should not be modified.

Procedure: How to Specify the Width of the Editor

The width of the editor depends on several factors. If the file exists, and no width is specified on the IEDIT command or as a GLOBALV parameter, the LRECL of the file is honored. You can specify a width greater than the width of the file in two ways:

- In the IEDIT command

  ```
  IEDIT fn ( WIDTH nnnn [any_valid_XEDIT_option]
  ```

- with the GLOBALV setting:

  ```
  GLOBALV SELECT IEDIT {SET|SETP} WIDTH nnnn
  ```

  You can issue the GLOBALV command from CMS or from FOCUS (prefaced by 'CMS'). IEDIT then uses this width as the default width unless it is changed. Using SET in the GLOBALV command causes the setting to remain in effect for the z/VM session. Using SETP makes the setting permanent, and lasts across z/VM sessions.

These widths are only honored if the WIDTH specified is greater than the width of the existing file. For files that do not exist, the WIDTH in the IEDIT command or the WIDTH in the GLOBALV setting is honored. If no width is given, and the file does not exist, it is opened as Fixed 80.

You can change the attributes of a file by issuing commands in XEDIT. For example:

```
RECFM V
```

However, you can only change the LRECL within the maximum width set for IEDIT, by default 256.

Using IEDIT on CMS

Most XEDIT features are available when you access XEDIT using the IEDIT command.

RIGHTP, LEFTP, RUN, SPVH, and SPHV are some of the commands that work differently when you issue the IEDIT command and the TED command. Many of these differences are due to the XEDIT environment.

- SPLITV and SPV split the screen into two vertical screens.
- SPLITH and SPH split the screen into two horizontal screens.
- SPVH and SPHV both split the IEDIT screen into four virtual screens.
- PF1/PF13 uses CMS HELP to display the IEDIT HELP screen. TED uses the TED editor to display the TED help screen.

When you have opened a file using the IEDIT command, you can open another file using the XEDIT command from the command line. The RUN command is still available.

Using IEDIT on z/OS

On z/OS, 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.

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.

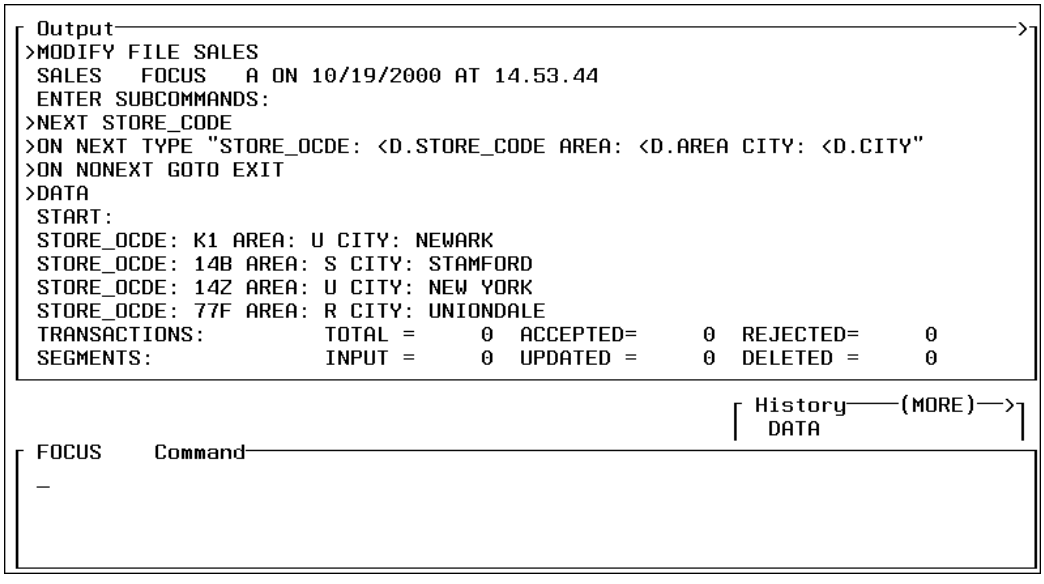
In this environment, your screen is divided into work areas called windows. Up to seven windows may appear on the screen at once. Each window accepts a specific type of user activity or performs a task.

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

Topics:

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

How to:

Enter 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 ?. In addition, the line end character in CMS (usually #) used to separate lines of input is not supported.

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 106 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

In this section:

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 97).
Output	Displays Command Window input and resulting output; accesses Hot Screen facility (see *Output Window* on page 100).
History	Lists commands and requests entered in the Command Window (see *History Window* on page 100).
Help	Displays function key settings. You may redefine settings for the current session (see *Help Window: Revising PF Key Settings* on page 101).
Table	Displays the most recent TABLE request (see *Table Window* on page 102).
Error	Displays FOCUS error messages (see *Error Window* on page 103).
Fields	Displays a list of available fields for a specified data source (see *Fields Window* on page 103).

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.

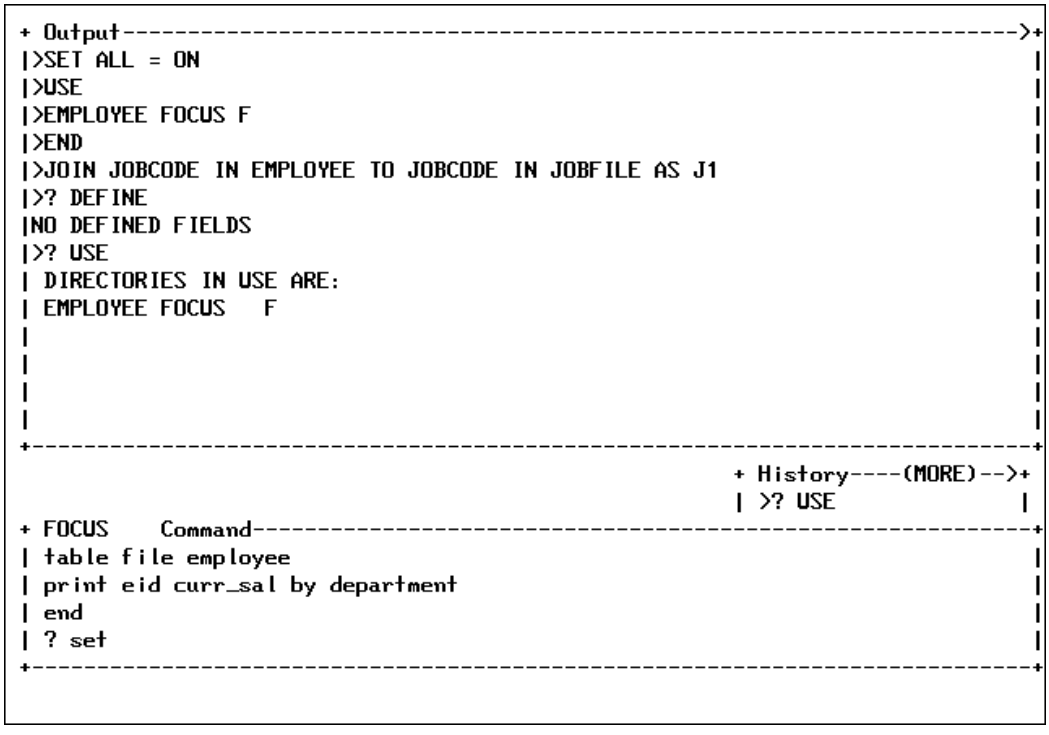
If your request is longer than four lines, press PF2 to enlarge the window. Finish entering the rest of the request; do not press PF2 again. Press Enter. The complete request is copied to the Output Window and to the History Window and is submitted for execution.

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.

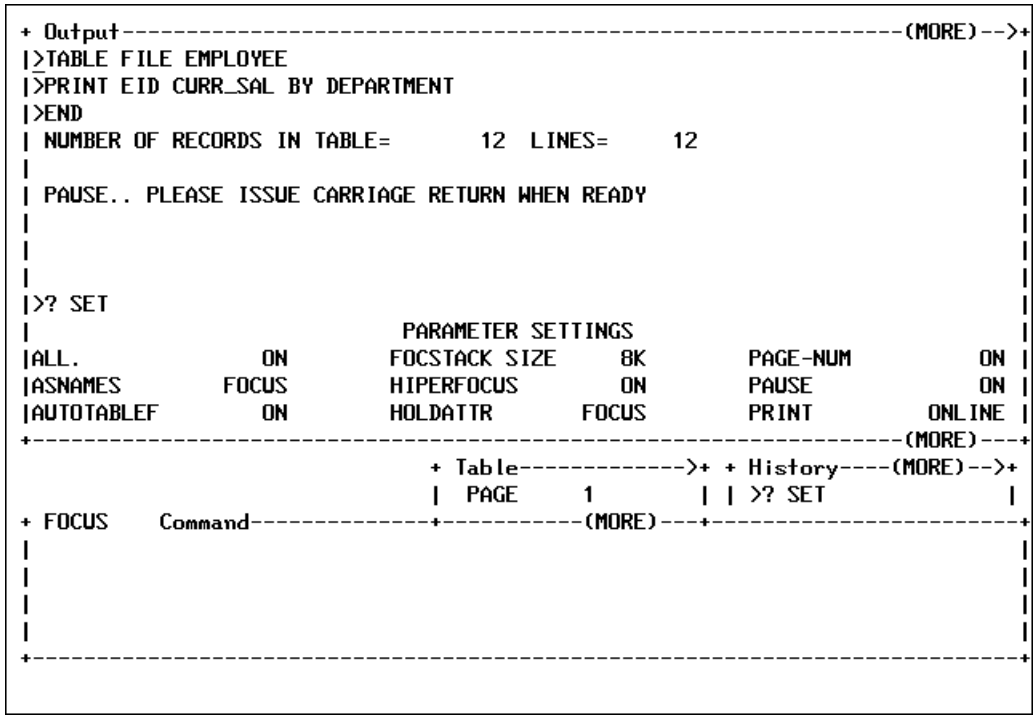
If you enter four FOCUS commands (each on a separate line) or a command and request (totaling four lines), the first item (command or request) is copied to the appropriate windows, submitted, and processed and followed by the next item. The Output and History Windows display each item in succession.

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 Hot Screen facility displays the report. When you press Enter and return from Hot Screen 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 Hot Screen displays the report.



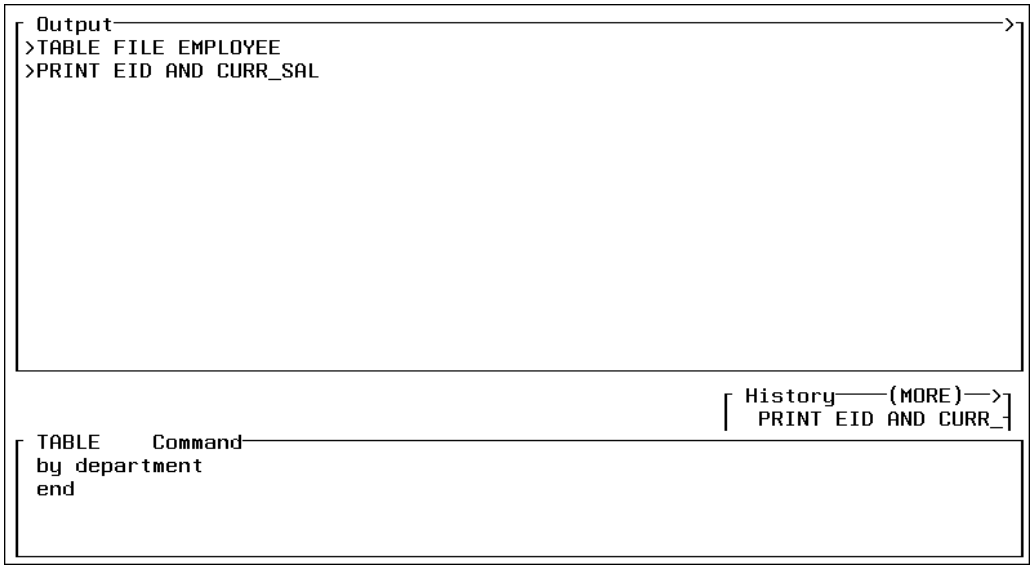
The Command Window also displays the current command mode in a title area. The title area is on the left side of the window's top border. It functions like the prompt does in the default FOCUS environment. In the previous screen sample, for example, the title area displays

```
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 Hot Screen 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 Hot Screen. Press Enter again to exit Hot Screen and return to the Terminal Operator Environment.

The Output Window does not log the report as displayed in the Hot Screen facility; it records only the report request. The Table Window contains the most recent report (see *Table Window* on page 102).

Within a TABLE or GRAPH request, you can use the Output Window to display field names, aliases, and formats for the specified data source (see *Fields Window* on page 103).

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 114.

Help Window: Revising PF Key Settings

Example:

Erasing the Window Contents

Assigning WINDOW Commands as 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 Hot Screen. 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.

  The correction appears in the Output Window after the line containing the error; in the History Window, an asterisk indicates the error, followed by the correction.

- 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.

Fields Window

The Fields Window appears when you issue the ?F query command from within a request. The Fields Window provides a list of fields for the specified data source.

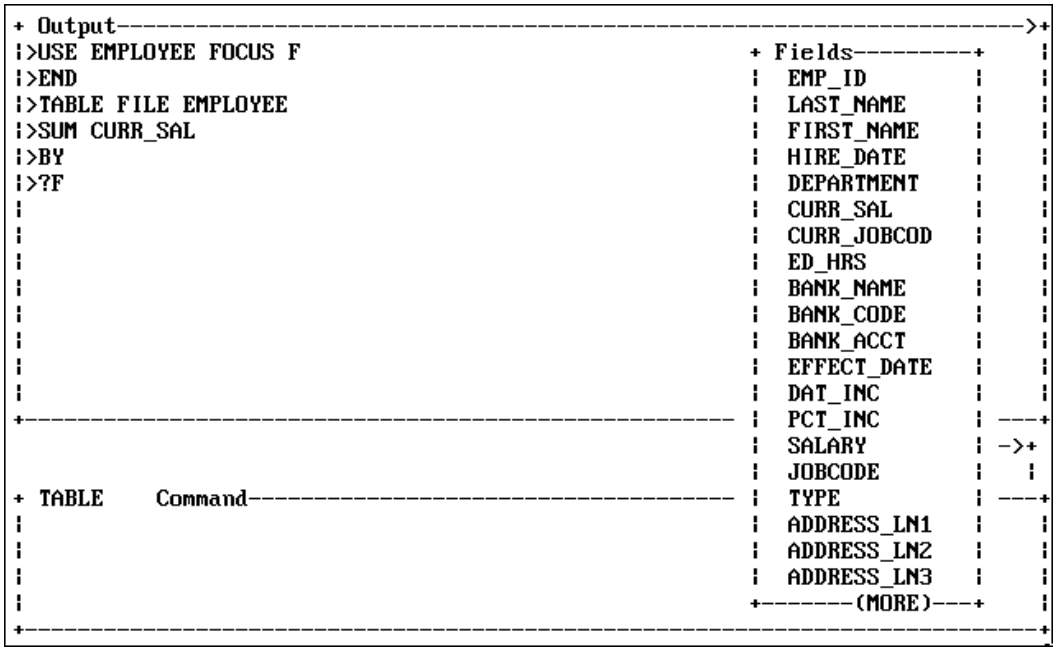
As you enter your request from the Command Window, issue the ?F query command:

```
?F
```

The Fields Window opens, displaying the active fields. Move the cursor next to the appropriate field. Press Enter. The Fields Window closes and the field appears in the Command Window.

To leave the Fields Window without making a selection, press PF12.

In the sample screen, part of a TABLE request has been entered and appears in the Output Window. The ?F query command has also been issued. As a result, the Fields Window overlays the Output and Command Windows. At this point, you may select a field or leave the window by pressing the PF12 key.



Note: If you have not issued a partial request and entered the ?F query command, the fields appear in the Output Window and are not available for selection.

Displaying Fields and Field Formats

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

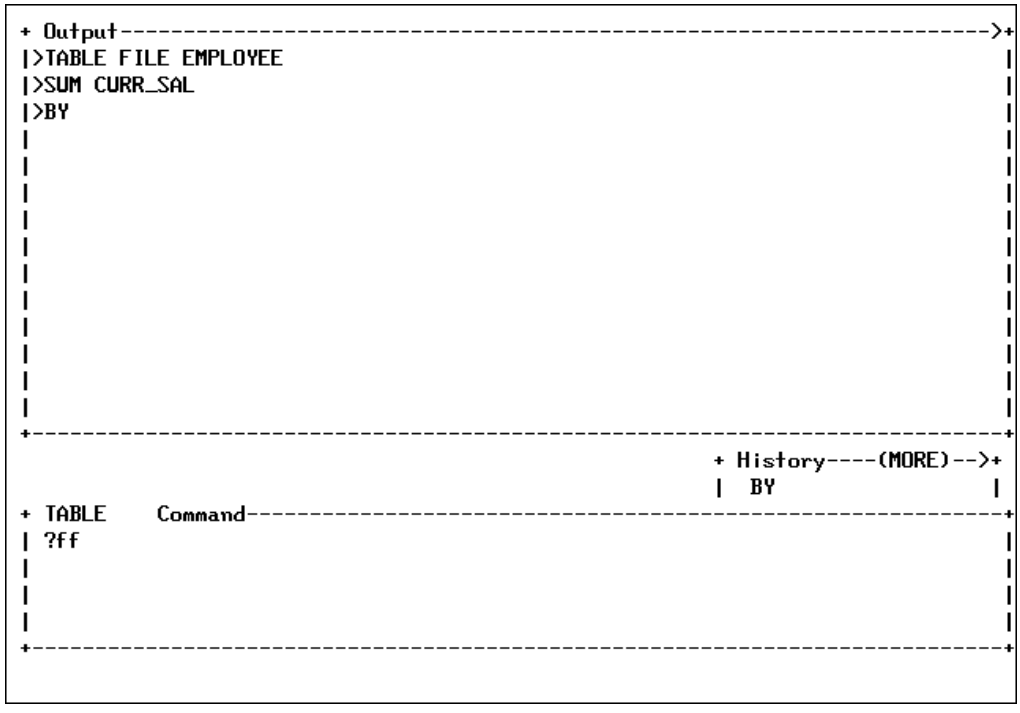
As you enter your request from the Command Window, issue the ?FF query command:

```
?FF
```

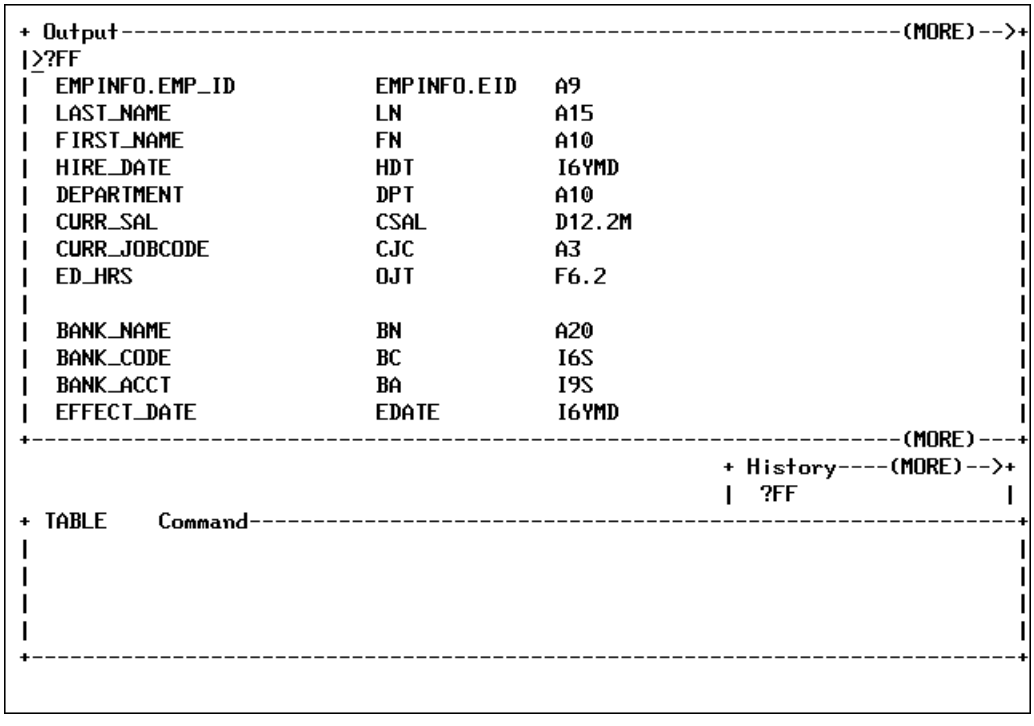
The Output Window displays the fields. Note the field you wish to use in the request and then press the PF12 key to return to the Command Window. Type the field you selected and press Enter.

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 sample screen, part of a TABLE request has been entered and appears in the Output Window. The ?FF query command has been typed into the Command Window.



When the Enter key is pressed, the Output Window displays the list of fields.



Window Commands

In this section:

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 101.) 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 file for z/VM CMS or to a data set for z/OS.
- ❏ 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, the Error Window, or the Fields Window.

Commands for Activating a Window

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

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

How to:

Erase the Contents of 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

How to:

Automatically Scroll the Contents of the Output Window

Control Data Transmission

Control Buffering of Line Mode Output

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}
```

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|OFF}
```

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

How to:

Remove a Window From the Screen

Display a Closed Window in its Normal Screen Location

Move a Window to a New Screen Location

Control the Length of an Error Message in the Error Window

Redefine Key Settings

Change the Height or Width of a Window

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 101).
- 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

How to:

Display 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

How to:

Enlarge 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

How to:

Route 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 a FILEDEF or ALLOCATE command before you issue the ROUTE command; space allocation is not set dynamically. In CMS, the ddname should define (FILEDEF) to a file with LRECL 132 and RECFM F. In TSO, the ddname should be allocated to a sequential data set with LRECL 132 and RECFM F.

Scrolling Window Contents

How to:

Scroll 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.

5 CMS Guide to Operations

As a CMS 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 CMS environment.

All of the FOCUS features described in this documentation set are available to you.

Release statistics, installation and operational changes, and maintenance log information (such as program temporary fix [PTF] information and release notes) are available online. To view the online information, issue:

```
EX READMEF
```

from the FOCUS prompt. After you execute the FOCEXEC, a menu appears with a choice of reports regarding release specific information. READMEF includes operational notes, installation notes, known problems, problems corrected in this release, new features, advisories, and license management information.

Topics:

- ❏ Referencing Files
- ❏ Application Files
- ❏ Extract Files
- ❏ Work Files
- ❏ FOCUS Facilities Under CMS

Referencing Files

In this section:

Defining Files

Dynamically Defining Files

How to:

Reference Files in FOCUS

Reference:

FOCUS Files

In FOCUS, you always refer to a CMS file by its fileid, which consists of the name of the file, the type of file, and the disk on which the file resides.

Syntax: How to Reference Files in FOCUS

filename filetype filemode

where:

filename

Is the name of the file.

filetype

Is the type of file. Filetype usually adheres to standard naming conventions.

filemode

Identifies the disk on which the file resides.

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 file name and are divided into categories:

System Files	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 commands.

System Files	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.
FOCUS	FOCUS data sources and external indices.
FOCSTYLE	FOCUS StyleSheet files.
FUSELIB	Library of user-written subroutines.
FOCCOMP	Compiled procedures.
TTEDIT	TableTalk sessions.
FMU	Window files.
TRF	Documentation for window files or window transfer files.
External Files	External data files not in FOCUS format, including transaction files.
HOLDSTAT Files	Documentation and DBA information for extract files.
WINFORMS	Winforms used in a Maintain procedure.

Extract Files	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.

Note: Dialogue Manager output files must be defined by you.

Work Files	Description
FOCSTACK	Used by Dialogue Manager to store FOCUS commands.
FOCSORT	Used during sorting.
FOCSML	A work area used by Financial Modeling Language.
*FOCPOST	Sequential output file saved using the POST or PICKUP commands.
REBUILD	Used by the REBUILD utility.
*EQFILE	Used for equation output by ANALYSE.
TABLTALK	Used by TableTalk as a procedure.

*The AS phrase renames asterisked files to allow more than one file of that type in a single session.

Defining Files

How to:

Define a File and Location

You can explicitly define files and their locations to FOCUS using the CMS FILEDEF command.

Syntax: How to Define a File and Location

```
FILEDEF ddname DISK filename filetype filemode (LRECL lrecl
RECFM recfm BLKSIZE blksize)
```

where:

ddname

Is the name used to refer to the file in FOCUS.

filename

Is the name under which the data is stored and is usually the same as *ddname*.

filetype

Identifies the type of file.

filemode

Identifies the disk.

Additionally, FOCUS will dynamically define most work and permanent files to the operating system during a FOCUS session.

There are other forms of the FILEDEF command. For details, see the IBM manual *CMS Command Reference.*

Dynamically Defining Files

How to:
Enter FOCUS
Exit FOCUS

You do not have to explicitly define all of your files prior to using to them in a FOCUS session. FOCUS will dynamically define certain files.

FOCUS will define some or all of the following output or work files during a FOCUS session:

- ❏ HOLD, SAVB and SAVE files.
- ❏ FOCUS data sources.
- ❏ FOCUS work files, such as FOCSTACK and FOCSORT.
- ❏ Financial Modeling Language file FOCSML.
- ❏ OFFLINE, SYSIN, and SYSPRINT files.
- ❏ TTEDIT files.
- ❏ FMU and TRF files (documentation only).
- ❏ WINFORMS files.

Syntax: How to Enter FOCUS

To enter FOCUS from CMS, execute the FOCUS EXEC

```
[EX] FOCUS [(NOPROF|(PROFILE filename]
```

where:

NOPROF

Is an optional parameter. Enables you to bypass or ignore an existing PROFILE FOCEXEC procedure when you enter your FOCUS session.

filename

Is an optional parameter. Is the name of an alternative PROFILE FOCEXEC which executes instead of the existing PROFILE FOCEXEC.

Syntax: How to Exit FOCUS

To exit FOCUS and return to CMS, enter the FIN command:

```
FIN
```

Application Files

In this section:

The following topics describe how FOCUS references and searches for your application files such as Master Files, FOCEXEC files (stored procedures), FOCUS data sources, FOCCOMP files, and external data files. It also describes the various functions of the USE command.

When you do not change the default file types and the default file mode (A), you can enter FOCUS and prepare reports, or modify data without issuing any FILEDEF commands or other communication. When you rename the default file type of a FOCUS data source, or use a data source on a disk other than the A disk, you must issue the USE command when you enter FOCUS. (The USE command is discussed in the *Describing Data* manual.)

Master Files

Master Files have the file type MASTER and consist of parameter lists that describe data sources to FOCUS. The description of a FOCUS data source and all data sources it cross-references must be available whenever you refer to the data source. Generally, the description and the data source both reside on the the A disk, although descriptions do *not* have to be on the same disk as the data. In this way, many users may share the same set of Master Files, yet use different data sources.

The Master File and the data source it is describing usually have the same file name. 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 only work with Master Files that consist of fixed or variable-length records up to 160 bytes long.

Note: All disks are searched when a Master File is needed. That is, the standard CMS search order is used (A then B, and so on) until the Master File is found or no more disks exist to be searched.

Access Files

Access Files for FOCUS data sources have the file type ACCESS. They are optional except for intelligent partitioning of FOCUS data sources. For information, see the *Describing Data* manual. 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.

Note: All disks are searched when an Access File is needed. That is, the standard CMS search order is used (A then B, and so on) until the Access File is found or no more disks exist to be searched.

FOCEXEC or Maintain Files

How to:

Execute a FOCEXEC Stored Procedure

Execute a Non-FOCEXEC Stored Procedure

Example:

Executing a Non-FOCEXEC Stored Procedure

Stored procedures can be saved under any CMS file name and file type. They are most conveniently filed under the file type FOCEXEC. Maintain procedures can conveniently have FILETYPE MAINTAIN. For details, see the *Maintaining Databases* 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 only work with FOCEXECs that consist of fixed or variable-length records up to 160 bytes long.

Syntax: How to Execute a FOCEXEC Stored Procedure

```
EXEC procedurename
```

or

```
EX procedurename
```

where:

procedurename

Is the name of the procedure to be executed. This corresponds to the file name.

Syntax: How to Execute a Non-FOCEXEC Stored Procedure

If you do not use the file type FOCEXEC, the full file name and file type must be enclosed in single quotation marks when you execute the procedure (since the identifier will contain an embedded blank). The syntax for this is:

```
EXEC 'procname proctype procmode'
```

or

```
EX 'procname proctype procmode'
```

where:

procname

Is the name of the procedure to be executed. This corresponds to the file name.

proctype

Is procedure type.

procmode

Is the procedure type. The default file mode is *. This means that the standard CMS disk search order is used (for example, A then B, and so on). Therefore, stored procedures may reside on central disks and be made simultaneously accessible to many users.

This type of identification can be useful when you need to group or sort many similar procedures.

In addition, users can execute a procedure from a specific minidisk by specifying:

```
EX 'filename filetype filemode'
```

Example: Executing a Non-FOCEXEC Stored Procedure

If a procedure is stored as

```
SALES REPORT A
```

To execute, enter:

```
EX 'SALES REPORT'
```

PROFILE FOCEXEC

In CMS, the PROFILE procedure must be named

```
PROFILE FOCEXEC A
```

unless the PROFILE option of the EX FOCUS command is used. (See *How to Enter FOCUS* on page 121 for more information.) Only the A disk or read-only extensions of the A disk (for example C/A) will be searched. The PROFILE will be executed before control is passed to the terminal.

StyleSheet Files

StyleSheet files can be saved under any CMS file name and have file type FOCSTYLE. The record format is fixed length with LRECL=80. For details, see the *Creating Reports* manual.

FOCUS Data Sources

FOCUS data sources contain data written in FOCUS format. FOCUS data sources have a record length of 4096 and a fixed-length record format. XFOCUS data sources have a record length of 16384 and a fixed-length record format. See the *Describing Data* manual for information about maximum file size and partitioning. Each data source has a file name that matches the name of its Master File, and has a file type of FOCUS. For example, if the data source's Master File is LEDGER MASTER, then the FOCUS data source is LEDGER FOCUS. You can override these defaults with the USE command, a DATASET attribute in the Master File, or an Access File. These techniques are explained in the *Describing Data* manual.

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 CMS, the external index is automatically defined as a permanent file when it is created using REBUILD.

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. In CMS, the MDI is automatically allocated as a permanent file when it is created using REBUILD.

Database Security: ENCRYPT, DECRYPT, and RESTRICT

How to:

Encrypt a FOCUS File

Decrypt a FOCUS File

Restrict a FOCUS File

Example:

Encrypting a FOCUS File

Restricting a FOCUS File

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 filename [filetype [filemode]]
```

where:

filename
Is the name of the FOCUS file.

filetype
Is the file type. The default is MASTER.

filemode
Is the file mode. The defaults is A.

Example: Encrypting a FOCUS File

The command

```
ENCRYPT FILE EMPLOYEE
```

encrypts the file EMPLOYEE MASTER A.

Syntax: How to Decrypt a FOCUS File

```
DECRYPT FILE filename [filetype [filemode]]
```

where:

filename
Is the name of the FOCUS file.

filetype
Is the file type. The default is MASTER.

filemode
Is the file mode. The defaults is A.

Syntax: How to Restrict a FOCUS File

```
RESTRICT
filename [filetype [filemode]]
filename [filetype [filemode]]
.
.
.
END
```

where:

filename

Is the name of the FOCUS file.

filetype

Is the file type. The default is FOCUS.

filemode

Is the file mode. The defaults is A.

Example: Restricting a FOCUS File

The command

```
RESTRICT
EMPLOYEE
SALES
PROD FOCUS C
END
```

restricts the files EMPLOYEE FOCUS A, SALES FOCUS A, and PROD FOCUS C.

FUSELIB

How to:

Access the FUSELIB Library

The FOCUS User-Written Subroutine Library, FUSELIB, contains additional calculation and utility routines.

Syntax: How to Access the FUSELIB Library

Issue the following CMS command before attempting to use one of these routines:

```
GLOBAL TXTLIB FUSELIB
```

FOCCOMP Files

How to:

Create a FOCCOMP File

The FOCCOMP file contains the output from the COMPILE command. The FOCCOMP file is used to run the compiled MODIFY procedure:

```
RUN focexecname
```

Syntax: How to Create a FOCCOMP File

```
filename FOCCOMP
```

where:

filename

Is the CMS file name of the FOCEXEC file that was compiled with the COMPILE command. The procedure has a variable-length record format and a record length of 4092.

Window Files

How to:

Identify Compiled Window Files

Identify Window Transfer Files

Identify Window Documentation 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 automatically when a Window Painter user chooses the *Create a new file* option, or invokes Window Painter and specifies a window file which does not yet exist. 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.

  Compiled window files are created on the A disk. Once they are created, they can be moved to any other disk. They have a 4096-byte record length and a fixed-length record format.

- 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, CMS) 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 statement or edited by Window Painter, they must be compiled using the WINDOW COMPILE command.

  When window transfer files are created by mainframe FOCUS Window Painter, they are created on the A disk. When they are transferred from PC/FOCUS, they are transferred to the disk specified in the prior CMS FILEDEF command. Once they are created or transferred, they can be moved to any disk. Transfer files have an 80-byte record length and a fixed-length record format.

  The Document the file option of the Window Painter Utilities Menu also creates a file with a TRF file type. If you need to create both TRF files (a documentation file and a transfer file) for a given compiled window file, be sure to give the two TRF files different file names. Otherwise, the second TRF file will be appended to the first one.

- 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.

  Window documentation files are created on the A disk. Once they are created, they can be moved to any other disk. They have an 80-byte record length and a fixed-length record format.

  The Create a transfer file option of the Window Painter Utilities Menu also creates a file with a TRF file type. If you need to create both TRF files (a documentation file and a transfer file) for a given compiled window file, be sure to give the two TRF files different file names. Otherwise, the second TRF file will be appended to the first one.

Syntax: How to Identify Compiled Window Files

```
filename FMU
```

where:

filename

Is the name chosen by the user during the Window Painter session that creates the file, or else is the name specified in the WINDOW COMPILE command that creates the file.

Syntax: How to Identify Window Transfer Files

```
filename TRF
```

where:

filename

Is the name chosen by the user when the transfer file is created by mainframe FOCUS Window Painter, or the name specified in the CMS FILEDEF command when the transfer file is transferred from PC/FOCUS.

Syntax: How to Identify Window Documentation Files

```
filename TRF
```

where:

filename

Is the name chosen by the user when the documentation file is created by Window Painter.

Non-FOCUS Data Sources

How to:

Set the VSAM Addressing Mode

Query the Addressing Mode

Example:

Defining a VSAM File to the Operating System

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 database files; and Teradata, CA-DATACOM/DB, Oracle, and SQL tables.

Non-FOCUS data sources must be defined to the operating system. You can let FOCUS issue default FILEDEF commands for your data sources, provided the files are either comma-delimited (COM) or fixed-format (FIX) files, where every record is a fixed- length, 80-character record. If a comma-delimited or fixed-format file has a record length other than 80, you must issue a FILEDEF.

For VSAM (KSDS or ESDS) files, use the standard DLBL commands to filedef the files prior to entering FOCUS. VSAM files exist only on OS or DOS disks (not on CMS disks). The Master Catalog must be defined. The dsname must be the VSAM cluster name. Use the IDCAMS utility to obtain it. The CMS LISTFILE command cannot be used for this purpose.

A SET command is available to switch the AMODE of the FOCSAM Interface (which reads VSAM and flat files) to 24-bit addressing. The Interface runs in 31-bit mode by default, in order to take advantage of modern operating system architecture. By extension, the Interface also builds 31-bit addresses for VSAM buffers and ACBs. However, some external VSAM buffering packages run in 24-bit mode, and do not recognize 31-bit addresses. The SET AMODE command allows the Interface to be run with these 24-bit programs.

Syntax: How to Set the VSAM Addressing Mode

```
{MVS|CMS} VSAM SET AMODE {24|31}
```

where:

`MVS|CMS`

Specifies the operating system.

`24`

Specifies AMODE 24. FOCSAM builds ACBs and buffers in 24-bit addresses.

`31`

Specifies AMODE 31. FOCSAM builds ACBs and buffers in 31-bit addresses. 31 is the default value.

Syntax: How to Query the Addressing Mode

To determine the addressing mode that is in effect at any time, you can issue the query

```
{MVS|CMS} VSAM SET ?
```

This query returns output similar to the following:

```
(FOC1177)  SET OPTIONS - : BUFND = n / BUFNI = n / AMODE = n
```

If you are not using any external programs or buffering packages that require 24-bit addresses for the ACB or buffers, you will not need to change the default.

Example: Defining a VSAM File to the Operating System

The following defines a VSAM file to CMS:

```
LINK VSAMDISK 191 192 R
ACC 192 B
DLBL IJSYSCT B DSN MASTER CAT (VSAM PERM)
DLBL CUST B DSN CUST DATA (VSAM PERM)
```

The allocation statements for a VSAM alternate index on CMS are:

```
DLBL CUST B DSN CUST DATA  (VSAM PERM
DLBL DD1  B DSN CUST PATH1 (VSAM PERM
DLBL DD2  B DSN CUST PATH2 (VSAM PERM
```

TRACE Files

How to:

Define a Trace File

Example:

Defining the Trace File

To record output from either the TRACE or ECHO option of MODIFY in a file, you must first define the file.

Syntax: How to Define a Trace File

```
FILEDEF HLIPRINT DISK fileid (RECFM recfm LRECL lrecl
```

where:

fileid
: Is the CMS fileid of the file receiving the TRACE or ECHO output.

recfm
: Is the format of the file (F for fixed, V for variable).

lrecl
: Is the file record length. The record length should be at least 32 for the TRACE option and at least 80 for the ECHO option.

If you are displaying output from the TRACE or ECHO option on the terminal, no allocation is necessary.

Example: Defining the Trace File

This request stores output from the TRACE option in the file TRACE OUTPUT A:

```
CMS FILEDEF HLIPRINT DISK TRACE OUTPUT (RECFM F LRECL 32)
MODIFY FILE EMPLOYEE TRACE
PROMPT EMP_ID CURR_SAL
IF CURR_SAL GT 50000 GOTO HIGHSAL
ELSE GOTO UPDATE;
```

```
CASE UPDATE
MATCH EMP_ID
   ON MATCH UPDATE CURR_SAL
   ON NOMATCH REJECT
ENDCASE

CASE HIGHSAL
TYPE
   " "
   "YOU ENTERED A SALARY ABOVE $50,000"
   " "
PROMPT CURR_SAL.PLEASE REENTER THE SALARY BELOW.
IF CURR_SAL GT 50000 GOTO HIGHSAL
ELSE GOTO UPDATE;
ENDCASE
DATA
```

TTEDIT Files

How to:

Edit a Saved TableTalk Session

Reference:

TTEDIT File ID

When you save a TableTalk session, a command file and a session file are saved. The command file has file type FOCEXEC and the session file has file type TTEDIT. Both files are created on the A disk. Each file has a fixed- length record format and a record length of 80. You can also edit a previous TableTalk session.

Reference: TTEDIT File ID

```
filename FOCEXEC A
filename TTEDIT A
```

where:

filename

Is the CMS file name you specify in TableTalk.

Syntax: How to Edit a Saved TableTalk Session

```
TABLETALK EDIT [filename]
```

where:

filename

Is the CMS 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

How to:

Specify a HOLDSTAT File

Reference:

Rules for Creating a HOLDSTAT File

HOLDSTAT files enable you to include FOCUS DBA information and environmental comments in HOLD and PCHOLD Master Files. You may create your own and specify its file name with the SET HOLDSTAT command or use the HOLDSTAT ERRORS file supplied by Information Builders.

The contents of a HOLDSTAT file are included automatically into HOLD and PCHOLD Master Files when report requests are executed. To include its contents, a HOLDSTAT file must be available and the SET HOLDSTAT command must be specified. For information about the SET HOLDSTAT command, see the *Developing Applications* manual.

Syntax: How to Specify a HOLDSTAT File

The fileids for a HOLDSTAT file are

```
{HOLDSTAT|fn} [MASTER|ERRORS] [fm|A]
```

where:

fn

Is the file name of your customized HOLDSTAT file.

MASTER

Specifies the MASTER file type. When FOCUS searches for the HOLDSTAT file, the MASTER file type takes precedence over the ERRORS file type.

ERRORS

Specifies the ERRORS file type.

fm

Is the file mode of the disk.

A HOLDSTAT file may contain environmental comments like a file header, or the FOCUS DBA attribute, or both. The supplied HOLDSTAT ERRORS file 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: Rules 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 attribute, see the *Describing Data* manual.

Winform Files

All of a Maintain procedure's Winforms are contained in a file with the file type WINFORMS. A Maintain procedure's WINFORMS and FOCEXEC files must have the same file name. 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 only work with WINFORMs that consist of fixed or variable-length records up to 160 bytes long. When searching for WINFORMS files, FOCUS uses the standard CMS search order.

Users create and edit Winforms using the Winform Painter. Users should make changes to WINFORMS files using the Painter only, and should not attempt to edit them directly. All changes made outside the Painter are lost the next time the file is edited in the Painter.

Extract Files

In this section:

Extract files save lines of user input and output during a FOCUS session.

By default, extract files are written to the z/VM minidisk specified by the SET TEMP command. If you do not issue the SET TEMP command, extract files are written to the minidisk with the largest amount of unused space to which you have write access. The name of an extract file is the AS name specified in the command that creates it or, if no AS name is specified, a default name (HOLD, SAVE, or SAVB). The file type is assigned based on the extract file format.

You use a FILEDEF command to assign a file name, file type, and file mode for an extract file.

In the case of a HOLD file, the Master File is not affected by the FILEDEF command. The Master File is written to the minidisk specified by the SET TEMP command, and its name is taken from the AS name in the HOLD command. If the HOLD command does not contain an AS phrase, the Master File name is HOLD.

Locating Extract Files

How to:

Reference:

Example:

As soon as FOCUS is entered, it searches for the disk with the largest available space that you have write access to and will place all work files there. The TEMP DISK parameter may be used to specify a particular disk for the creation of work and extract files. Changing the TEMP DISK will only affect the disk on which new files are created.

Example: Specifying a Location for Extract Files

The following places all extract files on the D disk:

```
SET TEMP DISK = D
```

Reference: Notes for Using the FILEDEF Command With an Extract File

The following should be considered when using the FILEDEF command:

- If a FOCSORT file is created, it is written to the minidisk specified by the SET TEMP command.
- The FILEDEF command must be in effect any time you use FOCUS to access the file.
- Do not specify DCB parameters for a HOLD file. If you do, they will be ignored.
- The FILEDEF command is not supported for creating extract files in FOCUS format or other DBMS formats.
- The FILEDEF command is supported for MATCH FILE requests.

Syntax: How to Issue a FILEDEF Command for Creating an Extract File

Issue the following command before creating the extract file

```
CMS FILEDEF ddname DISK filename filetype filemode
```

where:

ddname

Is the AS name from the HOLD, SAVE, or SAVB command. If the command did not specify an AS name, the *ddname* is HOLD, SAVE, or SAVB.

filename

Is the file name for the extract file.

filetype

Is the file type for the extract file.

filemode

Is the file mode for the extract file. You must have write access to this minidisk. If you do not have write access, the following error message is returned:

```
(FOC350) ERROR WRITING OUTPUT FILE: filename
```

Example: Using FILEDEF to Create a HOLD File

In the following examples, SET TEMP = T. The request is:

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY LAST_NAME BY FIRST_NAME
ON TABLE HOLD AS CURRSAL
END
```

Running this request with no FILEDEF command creates the following files:

```
CURRSAL  MASTER   T1
CURRSAL  FOCTEMP  T1
```

Issue the following FILEDEF command:

```
CMS FILEDEF CURRSAL DISK SALLIST DATA A
```

Now, running the same request creates the following files:

```
CURRSAL  MASTER   T1
SALLIST  DATA     A1
```

Note that the file name, file type, and destination minidisk for the extract file are taken from the FILEDEF command. The file name and destination minidisk for the Master File are not.

HOLD Files

How to:

Create a HOLD File

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.

If you specify the FORMAT FOCUS option with HOLD, FOCUS creates normal MASTER and FOCTEMP files, each with the file name FOC$HOLD, as well as the Master File for the final FOCUS data source. These files are then used as input to the MODIFY that creates the final FOCUS data source . The new FOCUS data source and Master File are created on your temporary disk (that is, the disk specified with the FOCUS command SET TEMP *disk*).

If you create a FOCUS data source larger than one gigabyte using HOLD FORMAT FOCUS, you will need a very large amount of TEMP space available.

Syntax: How to Create a HOLD File

```
HOLD [AS ddname]
```

where:

ddname

Is the name of the file that will contain the report output. The default name is HOLD. The fileid used is *ddname* FOCTEMP unless you specify the format option (see the *Creating Reports* manual). FOCUS also creates *ddname* MASTER to contain a Master File that describes the report output.

SAVB Files

How to:

Create a SAVB File

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 of the file. The default name is SAVB. The fileid is *ddname* FOCTEMP.

SAVE Files

How to:

Create a SAVE File

A SAVE file contains the results of a report request with all columns in the report in printable, character format with no spaces between columns.

Syntax: How to Create a SAVE File

```
SAVE [AS ddname]
```

where:

ddname

Is the name of the file. The default name is SAVE. The fileid is *ddname* FOCTEMP.

LOG and Transaction Files

How to:

Create a LOG File

Create a Transaction File

Send TYPE Messages to a Transaction File

Reference:

Dialogue Manager Input and Output Files

Example:

Logging Transactions to a LOG File

Reading Records From a Transaction File

Sending TYPE Messages to a Transaction File

LOG and transaction files contain FOCUS transactions.

Syntax: How to Create a LOG File

To log transactions on a file, first define the file with this command

```
FILEDEF ddname DISK fileid (RECFM recfm LRECL lrecl
```

where:

ddname

Is the *ddname* of the file specified by the LOG category ON *ddname* statement.

fileid

Is the CMS fileid of the file receiving the transactions.

recfm

Is the format of the file (F for fixed, V for variable).

lrecl

Is the file record length. The proper record length is discussed in the *Maintaining Databases* manual.

Example: Logging Transactions to a LOG File

For example, this request logs transactions in the file EMPTRANS TRANSACT A:

```
CMS FILEDEF ALLTRANS DISK EMPTRANS TRANSACT A (RECFM F LRECL 24
```

```
MODIFY FILE EMPLOYEE
LOG TRANS ON ALLTRANS
PROMPT EMP_ID CURR_SAL
VALIDATE
   SAL_TEST = IF CURR_SAL GT 50000 THEN 0 ELSE 1;
MATCH EMP_ID
   ON MATCH UPDATE CURR_SAL
   ON NOMATCH REJECT
DATA
```

Syntax: How to Create a Transaction File

To have a MODIFY request read a transaction file, define the file to the *ddname* specified in the FIXFORM, FREEFORM, or DATA statements by entering the CMS command

```
FILEDEF ddname DISK fileid (LRECL lrecl RECFM recfm BLKSIZE blksize)
```

where:

ddname

Is the *ddname* specified in the FIXFORM or DATA statement.

fileid

Is the CMS fileid of the transaction file.

When you create a transaction file to be read by the FIXFORM statement, be sure that it has a fixed-length record format (RECFM F).

You do not need to define the file if the file has a file type of DATA, a file mode of A and the records are 80-byte fixed format.

Example: Reading Records From a Transaction File

For example, this request reads fixed-format records from the file EMPFILE DATA A:

```
CMS FILEDEF FLOAFILE DISK EMPFILE DATA A

MODIFY FILE EMPLOYEE
COMPUTE FLOATSAL/F8=;
FIXFORM EMP_ID/12 FLOATSAL/F4
COMPUTE CURR_SAL = FLOATSAL;
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH UPDATE CURR_SAL
DATA ON FLOAFILE
END
```

Syntax: How to Send TYPE Messages to a Transaction File

To send MODIFY TYPE messages to a file, first define the file with this CMS command

```
FILEDEF ddname DISK fileid (RECFM recfm LRECL lrecl
```

where:

ddname

Is the *ddname* of the file specified by the TYPE ON *ddname* statement.

fileid

Is the CMS fileid of the file receiving the messages.

recfm

Is the format of the file (F for fixed, V for variable).

lrecl

Is the file record length.

Example: Sending TYPE Messages to a Transaction File

For example, this request records accepted transactions in the file ACCEPT TRANS A and rejected transactions in the file REJECT TRANS A:

```
CMS FILEDEF ACCFILE DISK ACCEPT TRANS A (RECFM F LRECL 80
CMS FILEDEF REJFILE DISK REJECT TRANS A (RECFM F LRECL 80

MODIFY FILE EMPLOYEE
PROMPT EMP_ID CURR_SAL
MATCH EMP_ID
   ON MATCH TYPE ON ACCFILE
      "<EMP_ID <12 <CURR_SAL"
   ON MATCH UPDATE CURR_SAL
   ON NOMATCH TYPE ON REJFILE
      "<EMP_ID <12 <CURR_SAL"
   ON NOMATCH REJECT
DATA
```

Reference: Dialogue Manager Input and Output Files

All files used in -READ or -WRITE statements must be explicitly defined.

Work Files

In this section:

Example:

In order to process certain types of requests, FOCUS needs to create temporary work files. As soon as FOCUS is entered, it searches for the disk with the largest available space that you have write access to and will place all work files there. The TEMP DISK parameter may be used to specify a particular disk for the creation of work and extract files. Changing the TEMP DISK will only affect the disk on which new files are created.

Example: Specifying a Location for Extract Files

The following places all extract files on the D disk:

```
SET TEMP DISK = D
```

FOCSTACK

Dialogue Manager uses a memory stack for work purposes with a default size of 8K bytes. However, if this is exceeded, FOCUS uses a work file with a fileid of FOCSTACK FOCTEMP.

FOCSORT

The TABLE, TABLEF, GRAPH, and MATCH commands may require a work file. This file is defined as FOCSORT FOCTEMP on the temporary disk. If the FOCSORT work file is defined to a disk other than the temporary disk, 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.

FOCSML

When Financial Modeling Language is used, a work area with the fileid FOCSML FOCTEMP is created.

FOCPOST

If Financial Modeling Language options POST or PICKUP are used, a sequential output file will be defined as FOCPOST FOCTEMP.

REBUILD

The REBUILD command options REBUILD and REORG require that a work file be defined under the *ddname* REBUILD. (See the *Maintaining Databases* manual for detailed information on the REBUILD command.) 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 sequential output file will be defined as

```
REBUILD FOCTEMP x4
```

where *x* is the letter of the disk with the greatest amount of free space to which you can write.

The INDEX option of REBUILD will use the sequential files SORTIN and SORTOUT. The REBUILD option, INDEX, requires an external sort package in the form of a TXTLIB. This sort package should be specified in a GLOBAL TXTLIB statement before REBUILD INDEX is used. See the *CMS Installation Guide* for further details.

When REBUILD INDEX is invoked, FOCUS searches all TXTLIBs specified in the current GLOBAL TXTLIB statement. If the sort is not found, FOCUS issues the following CMS command to search for the sort again:

```
GLOBAL TXTLIB sortlib
```

where *sortlib* is your library that contains sort routines. If the sort is still not located, the search is terminated and control is returned to the user with the message:

```
SORT COMPLETED. RETURN CODE 16
```

You can only REBUILD one partition of a partitioned data source at one time. You must explicitly define the file containing the partition to use in the REBUILD request. If you are rebuilding a data source larger than one gigabyte, you must have enough TEMP space available, or REBUILD in sections to a new file, which is the recommended technique.

EQFILE

This file is used for storing regression equations from ANALYSE. The default fileid is EQFILE FOCEXEC.

TABLTALK

This file is used for storing the procedure to be executed from within TableTalk. The fileid is TABLTALK FOCEXEC.

SET PRINT

Example:

Sending Output to a File

To send output to a file when the PRINT parameter is set to OFFLINE, define the file to *ddname* OFFLINE using a FILEDEF command after the offline has been closed with the OFFLINE CLOSE command.

Example: Sending Output to a File

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 CLOSE
FILEDEF OFFLINE DISK EMPL OUTPUT A (RECFM F LRECL 80
```

FOCUS Facilities Under CMS

In this section:

The following topics describe the use of FOCUS facilities in the CMS environment.

Using the LET Command

If you are in a CMS environment that defines the pound sign (#) to be the LINEND character and you are using FOCUS online, you cannot use pound signs in LET statements. This is because CP recognizes the pound sign as a LINEND symbol and passes the data to CMS as two separate lines, as if the Enter key had been pressed. You can solve this problem in two ways:

- Change the LINEND character to another character. To do this from FOCUS, enter:

  ```
  CMS CP TERM LINEND character
  ```

  where *character* is the new LINEND character. After you enter the LET statements, you may reset the LINEND character to the pound sign.

- Type the escape character (usually a double quote) before the pound sign. To do this, the CMS escape facility must be on. Enter:

  ```
  CMS CP TERM ESCAPE ON
  ```

  After you enter the LET command, you may set the escape facility off.

To query CMS on the status of your LINEND and escape characters, enter:

```
CMS CP QUERY TERM
```

If you are executing the LET statement from a FOCEXEC, you can use pound signs without changing the LINEND character or using the escape character. To save LET equivalences in a file, enter:

```
LET SAVE [filename]
```

The default file name is LETSAVE. The file type is FOCEXEC, and the file mode is A. For example, entering the command

```
LET SAVE EMPLET
```

saves the LET equivalences in the file EMPLET FOCEXEC A. Entering the command

```
LET SAVE
```

saves the LET equivalences in the file LETSAVE FOCEXEC A.

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.

Entering TED

How to:

Enter TED

Change a File Name in TED

Change a File Type in TED

Change a File Mode in TED

Issue CMS Commands From TED

Example:

Editing TED

Creating a TED Profile

You can edit a file with TED using the TED command or with your system editor using the IEDIT command.

Syntax: How to Enter TED

```
TED filename [filetype [filemode]]
```

where:

filename

Is the name of the file you want to create or edit.

filetype

Is the type of file you are creating or editing. There are a number of different types of files. Each file type specifies what the file is going to be used for. The table below illustrates some of them. The default file type is FOCEXEC.

filemode

Is the identifier of the disk that holds your file. If you do not specify a file mode, TED assumes a file mode of A1, which means that the file will become part of a file collection called your A disk (A1).

Example: Editing TED

Note the following examples:

File Class	File Type	Example
MASTER	MASTER	TED EXPERSON MASTER
ACCESS	ACCESS	TED EXPERSON ACCESS
FOCEXEC	FOCEXEC	TED EXPERSON
TEMPORARY file	FOCTEMP	TED SAVE FOCTEMP
DATA file	DATA	TED EXPERSON DATA

Note: The maximum record length supported by TED under CMS is 159. Also, files containing non-printable characters may not be displayed.

Example: Creating a TED Profile

When you enter TED, it searches all disks for a file PROFILE TED. If the file is found, it is executed before control passes to the user. A typical profile might be

```
1. SCALE ON
2. NUM ON
3. PF6 FILE
```

where:

1. Displays a scale on line 2 of the screen.
2. Sets the prefix area on and displays line numbers.
3. Redefines the PF6 key as FILE.

Syntax: How to Change a File Name in TED

```
FN newfilename
```

or

```
FILEName newfilename
```

where:

newfile
Is the new file name.

Syntax: How to Change a File Type in TED

```
FT newfiletype
```

or:

```
FILEType newfiletype
```

where:

newfiletype
Is the new file type.

Syntax: How to Change a File Mode in TED

```
FM newfilemode
```

or:

```
FILEMode newfilemode
```

where:

newfilemode
Is the new file mode.

Syntax: How to Issue CMS Commands From TED

```
CMS command
```

Note: Only certain CMS commands are allowed, since they are issued in CMS subset mode.

Using GRAPH

FOCUS uses the IBM program product Graphical Data Display Manager (GDDM) to create graphics on IBM high-resolution devices (see the *IBM GDDM Guide for Users*).

A GDDM stub must be linked into FOCUS before GDDM graphics may be used. This must be performed once as part of the FOCUS installation procedure. See the *CMS Installation Guide.*

To run color graphics on the IBM 3270-type devices, the GDDM library must be available at run time either as TXTLIBs specified in a GLOBAL statement or as a segment in memory.

You can use GDDM facilities for saving deferred graphic output. After creating the graph to be saved on your screen, press PF1 to save it. A prompt will appear asking for the name of the "save" file. Enter a name and press the Enter key. Thus saved, the file can be retrieved by compiling and linking the GDDM program ADMUSF2 (see the *IBM GDDM Guide for Users*).

Use the IBM-supplied utility ADMOPUV to print GDDM graphs on the IBM 3287 printer. ADMOPUV is run using the file and address of the printer as parameters (see the *IBM GDDM Guide for Users* for more information).

To get SAVE files and printouts from FOCUS GRAPH:

1. Generate a graph.
2. To get a printout, press PF4. Respond to the resulting prompt with the local printer name. The graph will be routed to the printer right away if the background print utility job (ADMOPUT) is active. If it is not active, you have to repeat the procedure with the PF4 key when the background job is running.

The ADMOPUV utility supplied with GDDM must be running in the background to take the printer output from GDDM to a local printer. Details about this utility can be found in the *IBM GDDM Guide for Users.*

It takes from five to fifteen minutes to print a graph depending on its complexity and colors. Keep in mind that the screen has six colors while the printer has only four, so colors (except for red, blue, or green) appear as black.

You can use FOCUS to generate graphs in conjunction with IBM's Interactive Chart Utility (ICU is a component of PGF). Specify normal FOCUS GRAPH syntax to use any of its features. The ICU Interface can either place the user directly in the ICU environment, or can save the graph format and data for subsequent ICU processing. To use the ICU Interface, issue the command:

```
SET DEVICE=ICU
```

Subsequent GRAPH requests will use ICU to generate graphs. See the *FOCUS ICU Interface User's Manual* for further information.

When allocating your own file for storing non-GDDM deferred graphic output, specify only the fileid and the DISP parameter. Do not specify the DCB parameters. FOCUS specifies them when it writes the file.

You can send the output of many GRAPH commands to one graphics SAVE file by setting DISP MOD. If you do, however, make sure that all of the deferred graphic output is headed for the same graphic device.

Accessing the FOCUS Menu

How to:

Access the FOCUS Menu

The FOCUS Menu is a convenient way to access FOCUS facilities using windows. It enables you to navigate through menu options, selecting FOCUS features such as the Talk technologies, DEFINE, JOIN, and query functions.

Syntax: How to Access the FOCUS Menu

Issue the command

```
EX FMMAIN
```

or

Invoke the FOCUS menu automatically when you enter FOCUS by editing the file SHELPROF FOCEXEC (found on either the FOCUS production disk or on any accessed disk). Remove the comment characters from the line with the -INCLUDE command. The lines should appear as follows:

```
-* This FOCEXEC will be executed only after PROFILE focexec execution
-* has been completed.
-INCLUDE FMMAIN
```

Accessing the FOCUS ToolKit

How to:

Access the FOCUS ToolKit

Example:

Dynamically Accessing Files Identified to the Toolkit

The FOCUS ToolKit is a menu-driven tool that walks the end user through many FOCUS facilities. Topics listed on the ToolKit menu include: reporting facilities, maintain data, decision support, and dictionary maintenance. The ToolKit option supports FOCUS and operating system utilities, as well as access to SQL facilities. An online guide to using the system is also included. The ToolKit option may be customized to provide specific site information, user selections for print destination, and database access.

The files ISHFPROF DATA (the background window) and ISHDPNTR DATA (printer selection list) must be available when you execute the ToolKit. These files are created with the installation FOCEXEC ISHFINSTL. The printer list may be updated using the FOCEXEC ISHFPNTR.

The ToolKit dynamically accesses files that are identified to the ToolKit. This is accomplished by creating a FOCEXEC that performs the necessary accesses and USE statements for a given file. This FOCEXEC is named *filename* ISHFALOC, where *filename* is the name of the file (as used by FOCUS).

Syntax: How to Access the FOCUS ToolKit

Issue the command

```
EX ISHFSHLL
```

or

Invoke the FOCUS ToolKit automatically when the user enters FOCUS by editing the file SHELPROF FOCEXEC, which is found on either the FOCUS production disk or on any accessed disk, to read:

```
EX ISHFSHLL
-EXIT
```

Example: Dynamically Accessing Files Identified to the Toolkit

For example, the file CAR ISHFALOC accesses the CAR FOCUS data source:

```
-CMS CP LINK CARDATA 191 201 RR
-CMS ACCESS 201 B
USE ADD
CAR FOCUS B
END
-EXIT
```

Accessing Power Reporter

How to:

Access Power Reporter

Power Reporter is a user-friendly, full screen front end to the FOCUS Report Writer. Designed for both end users and application developers, it features pull down menus that enable users to create FOCUS report requests in non-linear order, preview the report format and generated code, and much more. Power Reporter provides the capability to create DEFINEs and JOINs, save requests for later revision, and display information about the FOCUS session. Power Reporter has a PC Windows look and feel and contains context-sensitive help.

Syntax: How to Access Power Reporter

```
EX PWREP
```

National Language Support

How to:

Determine Current Language Settings

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 *CMS 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.

Issuing CMS Commands From Within FOCUS

Example:

Issuing a CMS Command From FOCUS

Valid CMS commands may be issued from within FOCUS by using the CMS prefix.

From within the SCAN facility, you can enter CMS followed by a valid CMS command on the same line. The line is passed to CMS for processing and resulting messages are from CMS.

All CMS commands that can execute in subset mode are valid; CMS itself decides which commands it will accept from inside FOCUS.

Valid CP commands should be prefixed with CMS CP. For example:

```
CMS CP QUERY TERM
```

Example: Issuing a CMS Command From FOCUS

The following issues the CMS LIST command:

```
CMS LIST * FOCEXEC A
```

Extended Plists

Example:

Issuing Commands With Extended Plist

Issuing Commands Without Extended Plist

Issuing Direct Operating System Commands From Within Dialogue Manager: The -CMS Command

z/VM FOCUS issues commands to CMS using CMS's Extended Plist. This enables FOCUS users to issue CMS commands such as STORMAP and PIPE that require the extended plist. It also enables FOCUS users to specify FILEDEF, ACCESS, and other CMS commands with parameters that are longer than eight characters. The standard plist or token plist uppercases each token, left-justifies it, and truncates it to a length of 8 bytes. Extended plist has no limit on the length of the plist passed.

FOCUS uses a FILEDEF for SYSIN to read from the terminal and from FOCEXECs. By default, FILEDEF uppercases all data. Reissuing FILEDEF with the LOWCASE option, allows a lowercase option list to be passed to z/VM. This requires all FOCUS commands to be typed in uppercase. This may be used for special cases when a lowercase plist is required.

Example: Issuing Commands With Extended Plist

Consider the following FILEDEF command:

```
FILEDEF MINE DSN TEST.SAMPLE.MAY
```

With extended plist, all of the data set name is preserved.

Example: Issuing Commands Without Extended Plist

Without extended plist, the command

```
FILEDEF MINE DSN TEST.SAMPLE.MAY
```

would read as:

```
FILEDEF
MINE
DSN
TEST.SAM
```

In this case, the data set name has lost the 'PLE.MAY' part of the qualifier. A tokenized plist is limited to 32 tokens, each of which may be between 1 and 8 bytes. The extended plist consists of a control block that points to an uppercase command token, a pointer to the start of the option list, and a pointer to the end of the option list. The option list is left unchanged.

Example: Issuing Direct Operating System Commands From Within Dialogue Manager: The -CMS Command

The Dialogue Manager statement -CMS is assumed to be an operating system control statement if the RUN statement is absent. It executes directly within Dialogue Manager. For instance:

```
-CMS FILEDEF MYINPUT DISK &FNAME DATA A1
```

After executing an operating system control statement, you can test the statistical variable &RETCODE to determine if it operated successfully. &RETCODE is the value returned by the operating system after an operating system statement executes.

If there is no error, the value of &RETCODE is returned as zero. The following example illustrates the usage of this test:

```
-CMS STATE MYTRANS DATA A1
-RUN
-IF &RETCODE NE 0 GOTO BAD ;
-CMS FILEDEF INDATA DISK MYTRANS DATA A1
MODIFY FILE SALES
.
.
.
DATA ON INDATA
END
-EXIT
-BAD TYPE TRANSACTION FILE NOT AVAILABLE
-EXIT
```

Note: Because FOCUS automatically executes all stacked commands whenever a statistical variable is encountered, the -RUN statement above is not necessary. However, we recommend that you include -RUN to make the procedure more readable.

Interrupting FOCUS

Reference:

Kill Execution: KX

Kill Typing: KT

Resume Typing: RT

Kill Execution: FX

Display Statistics: ?

When you are in the FOCUS command environment, you can issue an external interrupt to stop execution of some commands (MODIFY, SCAN, TABLE, TABLEF, MATCH, and GRAPH) when operating in line mode. This results in an orderly closing of the FOCUS data file and/or suppression of the output. Issue an interrupt, then type either KX, KT, RT, FX or ? and press Enter.

Note: To interrupt down to CP level, press PA1. Then, type BEGIN to continue the session or a CP command such as IPL CMS to start a new CMS session.

The effect of each interrupt is as follows:

KX	Kill execution, stay in FOCUS.
KT	Kill typing till 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 remain 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 file 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 CMS session. The interrupted command may terminate normally or it may be cut short. The FOCUS data file 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 FILE 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 file.

LOADLIBs Used by CMS FOCUS

How to:

Add and Delete GLOBAL Libraries

FOCUS adds required LOADLIBs to the GLOBAL library list automatically, in addition to other optional LOADLIBs. FOCUS accomplishes this through the use of two CMS EXECs: FOCADLIB and FOCDELIB. Before adding LOADLIBs to the GLOBAL list, the existing list is saved. Then FOCUS adds the required and optional LOADLIBs in front of any libraries you may have specified before entering FOCUS. Upon exiting FOCUS, the prior GLOBAL environment is restored to the list specified before entering FOCUS. The LOADLIBs added when entering FOCUS are removed. The EXECs FOCADLIB and FOCDELIB must be found in the CMS search sequence (A-Z) during a FOCUS session.

Prior entries can be retained in the GLOBAL list and new entries added by using the FOCADLIB EXEC. To delete entries while maintaining others in the list, use the FOCDELIB EXEC. For both FOCADLIB and FOCDELIB, the output from the EXEC is the return code of the GLOBAL command.

Syntax: How to Add and Delete GLOBAL Libraries

```
CMS EX {FOCADLIB|FOCDELIB} libtype lib1 [lib2 lib3...] [(QUIET ]
```

where:

`FOCADLIB`

Adds libraries to the beginning of the GLOBAL library list.

`FOCDELIB`

Deletes libraries from the GLOBAL library list.

`libtype`

Is a library of type LOADLIB, TXTLIB, MACLIB, etc.

`lib1...`

Are the names of the libraries to be added or deleted.

`QUIET`

Suppresses messages from the GLOBAL command. The open parenthesis is required.

Note: FUSELIB routines now reside in FUSELIB LOADLIB (rather than in a TXTLIB). Issuing GLOBAL TXTLIB FUSELIB still works because the TXTLIB still exists. However, CMS loads routines from the LOADLIB before searching the TXTLIBs.

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.

Release statistics, installation and operational changes, and maintenance log information (such as program temporary fix [PTF] information and release notes) are available online. The ERRORS and MASTER partitioned data sets must be allocated in order to run READMEF. To view the online information, issue:

```
EX READMEF
```

from the FOCUS prompt. After you execute the FOCEXEC, a menu appears with a choice of reports regarding release specific information. READMEF includes operational notes, installation notes, known problems corrected in this release, new features, advisories, and license management information.

Topics:

- ❏ Referencing Files
- ❏ Application Files
- ❏ Window Files
- ❏ Extract Files
- ❏ Work Files
- ❏ Calling FOCUS Under TSO
- ❏ FOCUS Facilities Under TSO
- ❏ TSO and FOCUS Interaction
- ❏ DYNAM Command

Referencing Files

In this section:

Allocating Files

Dynamically Allocating Files

Required Files

Reference:

FOCUS 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.

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.
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.

Application Files	Description
USERLIB	Library of user programs.
FUSELIB	Library of user-written subroutines.
FOCSTYLE	FOCUS StyleSheet files.
FOCCOMP	Compiled procedures.
ddname	Non-FOCUS data sources.
TTEDIT	TableTalk sessions.
FMU	Window files.
TRF	Documentation for window files or window transfer files.
HOLDSTAT Files	Documentation and DBA information for extract files.
WINFORMS	Winforms used in a MAINTAIN procedure.

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.
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
FOCSTACK	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.
*EQFILE	Used for equation output by ANALYSE.

Work Files	Description
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

Example:

Allocating Files using JCL

You can explicitly allocate files using JCL in your logon procedure or with TSO ALLOCATE 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 165 and *DYNAM Command* on page 223).

The JCL 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(WINFORMS)  DA(WINFORMS.DATA  SHR
ALLOC F(CAR)       DA(CAR.FOCUS)     SHR REUSE
```

Note: These allocation procedures are discussed in more detail in *Calling FOCUS Under TSO* on page 195.

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 FOCSTACK, 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
- ❏ FOCCOMP files
- ❏ ERRORS files
- ❏ Input data files
- ❏ TTEDIT files

- WINFORMS files

Note: *Required Files* on page 166 through *Work Files* on page 192 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 PDS 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 PDS 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(*)
  ```

  Once you enter FOCUS, you cannot reallocate SYSPRINT.

- 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 cannot reallocate SYSIN.

- 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 will not affect FOCUS output.

Application Files

In this section:

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, FOCCOMP files, 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 only work with files with an LRECL up to 160.

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 PDS must be referenced in the allocation, not a particular member.

The PDS 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 PDS.

Several partitioned data sets of FOCUS Master Files can be concatenated under the ddname MASTER, providing that they have identical LRECL and RECFM attributes.

Optionally, you can number the records in a Master File in positions 73-80. FOCUS distinguishes between numbered and unnumbered PDS 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. 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 only work with files with an LRECL up to 160.

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 PDS must be referenced in the allocation, not a particular member.

The PDS 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 PDS.

FOCEXEC or Maintain Files

How to:

Execute a Stored Procedure

Example:

Executing a Stored Procedure

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. Maintain procedures can be allocated to ddname Maintain. For more information, see *Maintaining Databases*.

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 only work with files with an LRECL up to 160. The line numbering conventions for Master Files apply to FOCEXEC files as well (see *Master Files* on page 168).

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 TSO, the procedure named PROFILE must be either a member of the PDS allocated to the ddname FOCEXEC or a separate file allocated to the ddname PROFILE. FOCUS checks the PDSs 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 195.

FOCEXECs as Sequential Files

Example:

Allocating 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.

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 is fixed length with LRECL=80. 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 PDS 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

How to:

Set FOCUS to Automatically Allocate FOCUS Data Sources

Example:

Allocating a FOCUS Data Source

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

How to:

Allocate a Multi-Volume Data Source in the z/OS Batch Environment

You have the option of allocating new FOCUS data sources, XFOCUS data sources, and FOCUS-created sequential files across multiple volumes.

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, TSO, and MSO 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=(TRK,(5,5))`
	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 be 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

How to:

Allocate Specific Volumes in TSO and z/OS FOCUS

Specify the Number of Units in TSO and z/OS FOCUS

Display the Volume Identifiers Allocated to a Multi-Volume Data Source

Example:

Allocating a Data Source to Two Volumes

Displaying Multi-Volume Data Set Information

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
FOCUSPAGES  =      1059
>  >
```

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

How to:

ENCRYPT a FOCUS File

DECRYPT a FOCUS File

RESTRICT a FOCUS File

Example:

Encrypting a FOCUS File

Restricting a FOCUS File

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 PDS to be encrypted.

Example: Encrypting a FOCUS File

The following command encrypts member EMPLOYEE of the PDS 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 PDS 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

Although random access techniques (BDAM) are used with FOCUS data sources, the files appear to the operating system as simple sequential data sets with a 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, IEHMOVE, 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 libraries: USERLIB and STEPLIB. USERLIB is searched first, if allocated. If the desired program is not found in the USERLIB library, then the call library, STEPLIB, JOBLIB, link pack area, and linklist are searched, in that order. This search logic is followed in all cases, whether the program to be loaded is an Interface module, a user-written exit from a computation expression, or a DATA VIA module.

USERLIB, if present, 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 or FUSELIB. 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.

FOCCOMP Files

How to:

Create a FOCCOMP File

The FOCCOMP file contains the output from the COMPILE command, and is used to run the compiled MODIFY procedure.

When you allocate this file, do not specify DCB attributes for this PDS. FOCUS chooses the most efficient block size for the disk type being used.

Note: You cannot use concatenated FOCCOMP data sets when compiling MODIFY procedures.

Syntax: How to Create a FOCCOMP File

```
COMPILE focexecname
```

where:

focexecname

Is the name of the FOCEXEC.

Window Files

In this section:

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 PDS. Before they can be created by Window Painter or by the WINDOW COMPILE command, a PDS must be created with LRECL 4096 and RECFM FB, and allocated to ddname FMU.

Once the PDS 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 PDS is not necessary if you are creating window files to be used only in the same FOCUS session. Window Painter will temporarily allocate the PDS.

Window Transfer Files and Window Documentation Files

Window transfer files and window documentation files are both members of the same PDS. Before they can be created by Window Painter, the PDS must be created with LRECL 80 to 132 and RECFM FB, and must be allocated to ddname TRF.

Once the PDS 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 PDS 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 PDS 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 PDS.

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 be appended to the older one in the PDS.

Non-FOCUS Data Sources

How to:

Set the VSAM Addressing Mode

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, Teradata, and Oracle 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 PDS 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.

A SET command is available to switch the AMODE of the FOCSAM Interface (which reads VSAM and flat files) to 24-bit addressing. The Interface runs in 31-bit mode by default, in order to take advantage of modern operating system architecture. By extension, the Interface also builds 31-bit addresses for VSAM buffers and ACBs. However, some external VSAM buffering packages run in 24-bit mode, and do not recognize 31-bit addresses. The SET AMODE command allows the Interface to be run with these 24-bit programs.

Syntax: How to Set the VSAM Addressing Mode

```
{MVS|CMS} VSAM SET AMODE {24|31}
```

With AMODE 31, FOCSAM builds ACBs and buffers in 31-bit addresses. 31 is the default. With AMODE 24, FOCSAM builds ACBs and buffers in 24-bit addresses.

To determine the addressing mode that is in effect at any time, you can issue the query

```
{MVS|CMS} VSAM SET ?
```

which returns the following output:

```
(FOC1177)   SET OPTIONS - : BUFND = n / BUFNI = n / AMODE = n
```

If you are not using any external programs or buffering packages that require 24-bit addresses for the ACB or buffers, you do not need to change the default.

TTEDIT Files

How to:

Edit a Saved TableTalk Session

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 PDS allocated to the 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 PDS allocated to the 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; however, the session will not be available for editing.

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 PDS to the ddname TTEDIT as OLD. An allocation for TTEDIT is not needed if the PDS '*prefix*.TTEDIT.DATA' exists. The PDS should have LRECL=80, BLKSIZE=1760, 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 CMS 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

How to:

Specify a HOLDSTAT File

Reference:

Considerations for Creating a HOLDSTAT File

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 PDSs.

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.

Winform Files

All of a Maintain procedure's Winforms are contained in a member of one or more partitioned data sets allocated to ddname WINFORMS. A Maintain procedure's WINFORMS and FOCEXEC files must have the same member name. 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 only work with files with an LRECL up to 160. When FOCUS needs to access a WINFORMS file, it searches for the ddname WINFORMS. If FOCUS finds that ddname WINFORMS is not allocated, it attempts to allocate the data set '*prefix*.WINFORMS.DATA' to ddname WINFORMS. If it does not find the specified member there, FOCUS displays an error message.

Users create and edit Winforms using the Winform Painter. Users should make changes to WINFORMS files using the Painter only, and not attempt to edit them directly. All changes made outside the Painter are lost the next time the file is edited in the Painter.

Extract Files

In this section:

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 191.

HOLD Files

How to:

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.)

Note:

- The HOLD FORMAT FOCUS option works by creating a normal HOLD file called FOC$HOLD, and then using that as input to the MODIFY that creates the required FOCUS data source. The Master File for the created FOCUS data source is stored in the PDS allocated to ddname HOLDMAST.
- If you create a FOCUS data source that is larger than one gigabyte using HOLD FORMAT FOCUS, you must explicitly allocate ddnames FOC$HOLD and FOCSORT to temporary files with enough space to hold the data.

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 PDS 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

How to:

Create a SAVB File

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 each for primary and 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 TSO COPY command.

SAVE Files

How to:

Create a SAVE File

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 PDS whose ddname is HOLDMAST. This PDS is exactly like a MASTER PDS except that FOCUS creates the members. It is usually a temporary file.

Note: If you are using an Interface 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 temporary data set with five primary and five secondary tracks.

Whenever FOCUS needs the description of a data source, it first searches the ddname MASTER. If the member is not found, it then searches the 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. If FOCEXEC is not allocated, the save is not performed. The default member name is LETSAVE.

- **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 173 for information.

Work Files

In this section:

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.

FOCSTACK

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 FOCSTACK as a temporary data set.

FOCSORT

The TABLE, TABLEF, GRAPH, and MATCH commands may require a sort work file. This is allocated under the ddname FOCSORT when required. If FOCSORT is allocated to a sequential file, FOCUS respects the allocation. To sort FOCUS data sources larger than one gigabyte, you must explicitly allocate ddname FOCSORT to a temporary file with enough space to hold the data.

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 temporary data sets 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
```

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
```

External Sort

The ddnames that FOCUS uses for external sort work files are of the form S001W*xxx*.

REBUILD

The REBUILD command options REBUILD and REORG require that a work file be allocated under the 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 may be allocated to disk or to tape. As disk files, they 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. If the file being rebuilt is larger than one gigabyte, you must explicitly allocate ddname REBUILD to a temporary file large enough to hold the data. 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
```

EQFILE

This file is used for storing regression equations from ANALYSE.

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.

Calling FOCUS Under TSO

In this section:

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

Example:

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 223 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

Example:

Entering FOCUS

Using CLISTs

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 the 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(WINFORMS) DA(WINFORMS.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 WINFORMS)
FREE  F(ISHFALOC PROFILE)
FREE  F(HOLD HOLDMAST SAVE REBUILD FOCSML FOCUS FOCSTACK)
FREE  F(FOCSORT FOCCOMP 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(WINFORMS)  DA('user.WINFORMS.DATA'             -
                      'prefix.WINFORMS.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 WINFORMS)
FREE  F(ISHFALOC PROFILE)
FREE  F(HOLD HOLDMAST SAVE REBUILD FOCSML FOCUS FOCSTACK)
FREE  F(FOCSORT FOCCOMP 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

In this section:

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

How to:

Enter TED

Issue TSO Commands From TED

Submit a Job From TED

Example:

Entering TED

Creating a TED Profile

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 Chapter 3, *Invoking Your System Editor With IEDIT*.

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 the ddname FOCEXEC. Specify only from the FOCUS command line.

FOCUS searches for the ddname of a sequential data set first. If a sequential file does not exist, FOCUS continues and searches for the member of a partitioned data set allocated to the ddname FOCEXEC. To change the search order and force FOCUS to search for the member first, issue:

```
TED FOCEXEC(member)
```

Example: Entering TED

You can specify a valid data set name or ddname with a member name if the file is a PDS. 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 the 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

```
{TSO|MVS} 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 160 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.

GRAPH

Reference:

Allocating Files to Hold Formatted Graphic Output

FOCUS uses the IBM program product Graphical Data Display Manager (GDDM) to create graphics on IBM high resolution devices (see list in the *IBM GDDM Guide for Users*).

A GDDM stub must be linked into FOCUS before GDDM graphics may be used. This must be performed once as part of the FOCUS installation procedure. See the *z/OS Installation Guide.*

To run color graphics on IBM 3270-type terminals, the GDDM load library must be allocated to ddname STEPLIB in your logon procedure.

You can use GDDM facilities for saving deferred graphic output. Before saving graphs, you must allocate ddname ADMSAVE as the PDS that will receive your graph(s). This can be allocated from within FOCUS as follows:

```
ATTRIB DCBG LRECL(400) RECFM(F) BLKSIZE(400)
ALLOC F(ADMSAVE) DA(ADMSAVE.DATA) DIR(5) SP(10 2) -
TRACKS USING(DCBG)
```

After creating the graph to be saved on your screen, press PF1 to save it. A prompt appears asking for the name of the “save” file. Enter a name and press the Enter key. The screen is saved in the ADMSAVE PDS as the member you have named. Thus saved, the file can be retrieved by compiling and linking the GDDM program ADMUSF2 (see the *IBM GDDM Guide for Users*).

Use the IBM-supplied utility ADMOPUV to print GDDM graphs on the IBM 3287 printer. ADMOPUV is run using the file and address of the printer as parameters (see the *IBM GDDM Guide for Users* for more information).

To get GDDM printouts from FOCUS GRAPH:

1. Generate a graph.
2. To get a printout, press PF4. Respond to the resulting prompt with the local printer name. The graph will be routed to the printer right away if the background print utility job (ADMOPUT) is active. If it is not active, you have to repeat the procedure with the PF4 key when the background job is running.

The ADMOPUV utility supplied with GDDM must be running in the background to take the printer output from GDDM to a local printer. Details about this utility can be found in the *IBM GDDM Guide for Users*.

It takes from five to fifteen minutes to print a graph depending on its complexity and colors. Keep in mind that the screen has six colors while the printer has only four, so colors (except for red, blue, or green) appear as black.

Reference: Allocating Files to Hold Formatted Graphic Output

When allocating your own file for storing non-GDDM deferred graphic output, specify only the data set name and the DISP parameter. Do not specify the DCB parameters. FOCUS specifies them when it writes the file.

If you let FOCUS allocate the SAVE file, you must copy it if you want it to remain after the end of the FOCUS session.

You can send the output of many GRAPH commands to one graphics SAVE file by setting DISP=MOD. If you do, however, make sure that all of the deferred graphic output is headed for the same graphic device.

Reference: Using the ICU Interface

You can use FOCUS to generate graphs in conjunction with IBM's Interactive Chart Utility (ICU is a component of PGF). Specify normal FOCUS GRAPH syntax to use any of its features. The ICU Interface can either place the user directly in the ICU environment or can save the graph format and data for subsequent ICU processing. To use the ICU Interface, issue the command:

```
SET DEVICE=ICU
```

See the *FOCUS ICU Interface User's Manual* for more information.

Accessing the FOCUS Menu

How to:

Access the FOCUS Menu

The FOCUS Menu is a convenient way to access FOCUS facilities using windows. It enables you to navigate through menu options, selecting FOCUS features such as the Talk Technologies, DEFINE, JOIN, and query functions.

For successful execution, the following minimal allocations are necessary:

```
DYNAM ALLOC FILE FOCEXEC DA prefix.FOCEXEC.DATA SHR REUSE
DYNAM ALLOC FILE ERRORS  DA prefix.ERRORS.DATA  SHR REUSE
DYNAM ALLOC FILE FMU     DA prefix.FMU.DATA     SHR REUSE
```

where *prefix* is the high-level qualifier selected for FOCUS installation at your site. Any other data sets may be concatenated to these ddnames.

Syntax: How to Access the FOCUS Menu

Issue the command

```
EX FMMAIN
```

or

Invoke the FOCUS menu automatically when you enter FOCUS by editing the member SHELPROF in the FOCEXEC.DATA partitioned data set (provided on the installation tape). Remove the comment characters from the line with the -INCLUDE command. The lines should appear as follows:

```
-* This FOCEXEC will be executed only after PROFILE focexec execution
-* has been completed.
-INCLUDE FMMAIN
```

Accessing the FOCUS ToolKit

How to:

Access the FOCUS ToolKit

Example:

Dynamically Accessing Files Identified to the Toolkit

The FOCUS ToolKit is a menu-driven tool that walks the end user through many FOCUS facilities. Topics listed on the ToolKit menu include: reporting facilities, maintain data, decision support, and dictionary maintenance. The ToolKit option supports FOCUS and operating system utilities, as well as access to SQL facilities. An online guide to using the system is also included. The ToolKit option may be customized to provide specific site information, user selections for print destination, and data access.

For successful execution, the following minimal allocations are necessary:

```
DYNAM ALLOC FILE FOCEXEC  DA prefix.FOCEXEC.DATA  SHR REUSE
DYNAM ALLOC FILE ERRORS   DA prefix.ERRORS.DATA   SHR REUSE
DYNAM ALLOC FILE FMU      DA prefix.FMU.DATA      SHR REUSE
DYNAM ALLOC FILE ISHFALOC DA prefix.ISHFALOC.DATA SHR REUSE
```

where *prefix* is the high-level qualifier selected for FOCUS installation at your site. Any other data sets may be concatenated to these ddnames. The members ISHFPROF (the background window) and ISHDPNTR (printer selection list) must be available in ddname ISHFALOC when you execute the ToolKit. These files are created with the installation FOCEXEC ISHFINSTL. The printer list may be updated using the FOCEXEC ISHFPNTR.

The ToolKit dynamically accesses files that are identified to the ToolKit. This is accomplished by creating a FOCEXEC that performs the necessary allocations and USE statements for a given file. This FOCEXEC is stored in the data set allocated to ddname ISHFALOC with a member name that corresponds with the file name (as used by FOCUS).

Syntax: How to Access the FOCUS ToolKit

Issue the command

```
EX ISHFSHLL
```

or

invoke the FOCUS ToolKit automatically when the user enters FOCUS by editing the file SHELPROF FOCEXEC, which is found on either the FOCUS production disk or on any accessed disk, to read:

```
EX ISHFSHLL
-EXIT
```

Example: Dynamically Accessing Files Identified to the Toolkit

For example, the member CAR accesses the CAR FOCUS data source:

```
DYNAM ALLOC FILE CAR DA prefix.CAR.FOCUS SHR REUSE
-EXIT
```

Accessing Power Reporter

How to:

Access Power Reporter

Power Reporter is a user-friendly full screen front end to the FOCUS Report Writer. Designed for both end users and application developers, it features pull down menus that enable users to create FOCUS report requests in non-linear order, preview the report format and generated code, and much more. Power Reporter includes capability to create DEFINEs and JOINs, save requests for later revision, and display information about the FOCUS session.

For successful execution, the following minimal allocations are necessary:

```
DYNAM ALLOC FILE FOCEXEC DA prefix.FOCEXEC.DATA SHR REUSE
DYNAM ALLOC FILE ERRORS  DA prefix.ERRORS.DATA  SHR REUSE
DYNAM ALLOC FILE FMU     DA prefix.FMU.DATA     SHR REUSE
```

where *prefix* is the high-level qualifier selected for FOCUS installation at your site. Any other data sets may be concatenated to these ddnames.

Syntax: How to Access Power Reporter

Issue the command:

```
EX PWREP
```

National Language Support

How to:

Determine Current Language Settings

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

In this section:

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

How to:

Issue a TSO Command From Within FOCUS

Example:

Executing a CLIST 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, like 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 (PDS) allocated to the 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(O,'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 SET MESSAGE command (in this case, 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 PDS was allocated. For example, the command TSO LIST INDATA(CAR), calls the TSO command LIST to display member CAR in the PDS 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 PDS allocated to ddname FOCEXEC, you enter:

```
TSO LIST FOCEXEC(ACCOUNTS)
```

FOCUS displays the member, ACCOUNTS, in the PDS 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

Reference:

Kill Execution: KX

Kill Typing: KT

Resume Typing: RT

Kill Execution: FX

Display Statistics: ?

Interrupting TSO Commands

TSO Time Check

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

Terminal	Function Key
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.

Reference: TSO Time Check

If you hit the interrupt key twice, you return to the TSO environment and you can enter the TSO command TIME. You can then resume your FOCUS session by pressing the Enter key. Entering any other TSO command leaves you in the TSO environment after the command is processed.

ISPF From FOCUS

How to:

Call a Procedure From Within 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 PDS (LRECL=80, RECFM=FB, BLKSIZE=1600) to contain your procedure.
2. Concatenate the PDS to your allocation for ddname SYSPROC.
3. Create a member in this PDS 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 PDS 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 PDS 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 PDS 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

How to:

Create a General List of Allocated Files

Determine If a Database Exists: ? TSO DSNAME

Estimate Data Set Sizes to Determine Available Space

Reference:

TSO ddname Variables

Example:

Listing the Currently Allocated Files

Displaying Attributes of a Queried ddname

TSO System Variables

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
WINFORMS              1         INPDQ.WINFORMS.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
FOCUSPAGES            =          8
```

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 221.

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).
&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.
&FOCUSPAGES	Posted only for FOCUS data sources. This number is the total number of 4096-byte pages in the data source. This is the same as the total number of pages shown by the ? FILE file name command. It is also the same as the highest page number shown by the ? FDT file name command. It does not include SHADOW pages and directory.

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:

```
Available pages (without additional extents) = BLKSWRITTEN -FOCUSPAGES
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:

```
(FOCUSPAGES x 2) + 3
```

DYNAM Command

In this section:

How to:

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 data set. 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).

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.

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

How to:

Allocate a Data Set With the ALLOCATE Subcommand

Example:

Using the DYNAM ALLOCATE Command

The DYNAM ALLOCATE command allocates a data set.

Syntax: How to Allocate a 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]

[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]
 [RECFM recfm] [REFDD ddname] [REFDSN dsname]
```

SMS and VSAM operands are:

```
 [DATACLASS name] [DSNTYPE {LIBRARY|PDS}]

 [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*
DD
DEFER

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.

`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.

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}

LIBRARY is for new partitioned extended (PDSE) and PDS is for new partitioned data set. A PDSE cannot contain load modules, should be SMS managed and allows concurrent updating of different members.

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
```

CONCAT Subcommand

How to:

Concatenate Data Sets

Example:

Using the DYNAM CONCAT Command

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

How to:

Deallocate Data Sets

Example:

Using the DYNAM FREE Command

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

How to:

Close Data Sets

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 237).

COPY Subcommand

How to:

Copy Entire z/OS Data Sets or Selected PDS Members

Example:

Using the DYNAM COPY Command

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.
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.

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

How to:

Copy Sequential Data Sets, PDS Members, or HiperFOCUS Files

Example:

Using the DYNAM COPYDD Command

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

How to:

Delete an Entire z/OS Data Set or Selected PDS Members

Example:

Using the DYNAM DELETE Command

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

How to:

Rename an Entire z/OS Data Set or Selected PDS Members

Example:

Using the DYNAM RENAME Command

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

How to:

Submit a Job to z/OS

Example:

Using the DYNAM SUBMIT Command

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

How to:

Compress a Partitioned Data Set

Example:

Using the DYNAM COMPRESS Command

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

Example:

Allocating an Existing File

Creating a New Data Set

Freeing Files

Concatenating Files

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
```

7 Recording and Replaying a FOCUS Session

The FOCREPLAY tool enables you to record your key strokes in an interactive FOCUS session and play it back in batch any time in the future.

Topics:

- ❏ Introduction to FOCREPLAY
- ❏ Configuring FOCREPLAY
- ❏ Recording a FOCUS Session
- ❏ Replaying a Recorded FOCUS Session
- ❏ Comparing a Recorded Session With a Replayed Session
- ❏ Stopping Replay at a Break Point
- ❏ Re-recording From the Middle or End of a Script

Introduction to FOCREPLAY

Reference:

Usage Notes and Prerequisites for FOCREPLAY

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. You invoke the FOCREPLAY functions by allocating the DDNAMEs FOCREPLAY uses for recording and playing back a FOCUS session.

The recording function of FOCREPLAY is called *input mode*. In this mode, you are in charge of the keyboard. It produces two outputs:

- ❏ A script that you can later use to replay the application.
- ❏ An output file containing the captured FOCUS session, including any screens displayed during the session.

To play back the script you invoke *replay mode*, which is activated by allocating a previously recorded script as the input to FOCREPLAY. Replay mode can be run in batch or interactively. During an interactive replay session, FOCREPLAY is in charge of the keyboard. It presents each line in the script and waits for a response from you in order to proceed. Replay mode produces an output file containing the replayed session.

You can then use any file comparison tool installed at your site to compare the original session's output to the replayed session's output.

One of the uses for this tool is to simplify acceptance testing of a new release of FOCUS. If you record an application's execution in your current release and then upgrade to a new release, you can play back the script and compare the resulting output file with the results of the original run to determine if the application produces the same output under the new release.

Reference: Usage Notes and Prerequisites for FOCREPLAY

- FOCREPLAY was tested with IBM's PC3270 emulator. However, it should work with all 3270 emulators such as EXTRA, Reflections, and Hummingbird if they use standard 3270 job stream calls.
- In order to judge the results of the comparison between the original output and the replay, you must understand any factors in the environment may be different and may affect the outcome. Some examples are expired fields, dates that have passed and are no longer valid, changes to any screen or menu, or the date and time of running the application.
- You should carefully document the conditions for running the script so that you can replicate these conditions accurately when replaying it.

  For example, assume your application contains a MODIFY or Maintain request that includes, updates, or deletes instances from a data source. In order to replay the session and get the same results, you need to run it against a copy of the data source as it was prior to the original run, and you need to run it against the transactions used in the original run.

  You also must keep a record of the operating system release and service level, database engine release, and FOCUS release.
- FOCREPLAY does not do the comparison between the original run and the replay. You can use any file comparison tool you have installed at your site, for example XCOMPARE on z/VM or ISPF 3.13 on z/OS.
- FOCREPLAY cannot be used with any feature that requires allocations outside of FOCUS (for example, FOCCALC or HLI) or with any feature that starts a service (such as MSO). Output of operating system commands is not captured.

Reference: Troubleshooting FOCREPLAY Applications

Before calling in a problem, make sure you have accounted for acceptable differences in the comparison between two releases. For example, when you run your application the second time and you want to compare its output to that of your initial (base) execution of the application, dates, release levels, and operating system messages will be different from the base. Remove these acceptable differences from the comparison between the two runs.

Configuring FOCREPLAY

How to:

Activate FOCREPLAY Input Mode

Issue FILEDEFs for Input Mode on z/VM

Allocate DDNAMEs for Input Mode on z/OS

Activate FOCREPLAY Replay Mode

Issue FILEDEFs for Replay Mode on z/VM

Allocate DDNAMEs for Replay Mode on z/OS

Example:

Sample FILEDEFs for Input Mode on z/VM

Allocating DDNAMEs for Input Mode on z/OS

Sample FILEDEFs for Replay Mode on z/VM

Allocating DDNAMEs for Replay Mode on z/OS

FOCREPLAY is activated by allocating the DDNAMEs it uses to record, capture, and/or replay a FOCUS session. You can add these allocations to your REXX EXEC, CLIST, or batch JCL that invokes FOCUS.

The required disposition for the FOCREPLAY data sets is DISP=MOD. The recommended DCB attributes are RECFM=VB, LRECL=233472, BLKSIZE=23476 (half track blocking), DSORG=PS. If you want the data sets to be members of a library, you can allocate them as members of a PDSE. Because standard PDS data sets do not support DISP=MOD allocation for their members, you cannot allocate the data sets as members of a PDS. If you want to store them in a PDS, you can allocate them as sequential files and copy them to PDS members after they are complete.

If the DCB attributes differ from the recommended values, they will be changed dynamically during the session. You must have read/write access to all of the FOCREPLAY data sets.

Note:

- You can make limited edits to a recorded script file. For example, you can insert a break at any point if you want to execute the script up to that point and then stop. The TED editor cannot edit these files. You must use a system editor such as XEDIT on z/VM or the ISPF editor on z/OS to edit them. You can access the system editor from FOCUS using the IEDIT command.
- The FOCREPLAY output files are created in binary mode with required blanks at the end of some of the records. In the ISPF editor, you must issue the PRESERVE ON command to retain these blanks.

Procedure: How to Activate FOCREPLAY Input Mode

FOCREPLAY records your FOCUS session if you allocate the following two DDNAMEs:

- **FSC3270I.** This DDNAME will contain the script of the session. You can play back this script to rerun the session by allocating the script file as input to replay mode. The 3270 terminal characteristics and session mode (height, width, graphic support, colors, and external attributes) are automatically recorded in this file. This enables you to record sessions for different types of terminals or different modes. Do not edit these lines.
- **FSC3270O.** This DDNAME will contain the captured FOCUS session.

If you allocate an existing file to either DDNAME, the new script and session output are appended to the existing information in the files.

Syntax: How to Issue FILEDEFs for Input Mode on z/VM

```
FILEDEF FSC3270I DISK script_file ft1 fm (DISP MOD
FILEDEF FSC3270O DISK session_output ft2 fm (DISP MOD
```

where:

script_file ft1 fm

Is the filename, filetype, and filemode of the file that will contain the script of the FOCUS session.

session_output ft2 fm

Is the filename, filetype, and filemode of the file that will contain the captured FOCUS session.

Example: Sample FILEDEFs for Input Mode on z/VM

```
FILEDEF FSC3270I DISK FSC3270 IN A (DISP MOD
FILEDEF FSC3270O DISK FSC3270 OUT A (DISP MOD
```

Syntax: How to Allocate DDNAMEs for Input Mode on z/OS

```
ALLOC F(FSC3270I) DA('hlq.script.file') MOD
ALLOC F(FSC3270O) DA('hlq.session.output') MOD
```

where:

hlq.script.file

Is the data set that will contain the script of the FOCUS session.

hlq.session.output

Is the data set that will contain the captured FOCUS session.

Example: Allocating DDNAMEs for Input Mode on z/OS

```
ALLOC F(FSC3270I) DA('USER1.FSC3270I.DATA') MOD
ALLOC F(FSC3270O) DA('USER1.FSC3270O.DATA') MOD
```

Procedure: How to Activate FOCREPLAY Replay Mode

FOCREPLAY replays your FOCUS session and creates a new output file containing the replayed session if you allocate the following three DDNAMEs:

- **FSC3270Q.** This DDNAME must be allocated to the script of the previously recorded session that you want to play back.
- **FSC3270O.** This DDNAME will contain the replayed FOCUS session. If you allocate the same file that was used when you recorded the session, the played back session will be appended to the recorded session. If you allocate a different output file, the played back session will be in a separate file from the original session.
- **FSC3270I.** This DDNAME should be allocated as DUMMY when replaying a session. If you allocate it to a file, you will both play back the session and record a new input script in the allocated file. This is necessary to use the <RECORD> function described later.

Syntax: How to Issue FILEDEFs for Replay Mode on z/VM

```
FILEDEF FSC3270Q DISK script_file ft1 fm (DISP MOD
FILEDEF FSC3270O DISK session2_output ft2 fm (DISP MOD
FILEDEF FSC3270I DUMMY
```

where:

script_file ft1 fm

Is the filename, filetype, and filemode of the file that contains the script of the previously recorded FOCUS session.

session2_output ft2 fm

Is the filename, filetype, and filemode of the file that will contain the replayed session.

Example: Sample FILEDEFs for Replay Mode on z/VM

```
FILEDEF FSC3270Q DISK FSC3270 IN A (DISP MOD
FILEDEF FSC3270O DISK FSC3270 OUT2 A (DISP MOD
FILEDEF FSC3270I DUMMY
```

Syntax: How to Allocate DDNAMEs for Replay Mode on z/OS

In a CLIST:

```
ALLOC F(FSC3270Q) DA('hlq.script.file') MOD
ALLOC F(FSC3270O) DA('hlq.session2.output') MOD
ALLOC (FSC3270I) DUMMY
```

In a batch job:

```
//FSC3270Q DD DISP=MOD,DSN=hlq.script.file
//FSC3270O DD DISP=MOD,DSN=hlq.session2.output
//FSC3270I DD DUMMY
```

where:

hlq.script.file

Is the data set that contains the script of the previously recorded FOCUS session.

hlq.session2.output

Is the data set that will contain the replayed session.

Example: Allocating DDNAMEs for Replay Mode on z/OS

In a CLIST:

```
ALLOC F(FSC3270Q) DA('USER1.FSC3270I.DATA') MOD
ALLOC F(FSC3270O) DA('USER1.FSC3270O.DATA2') MOD
ALLOC F(FSC3270I) DUMMY
```

In a batch job:

```
//FSC3270Q DD DISP=MOD,DSN=USER1.FSC3270I.DATA
//FSC3270O DD DISP=MOD,DSN=USER1.FSC3270O.DATA2
//FSC3270I DD DUMMY
```

Recording a FOCUS Session

This section illustrates how to record a FOCUS session under z/VM and z/OS.

Example: Recording a Session Under z/VM

Issue FILEDEF commands for the FOCREPLAY input mode DDNAMEs prior to invoking FOCUS or in the REXX EXEC that invokes FOCUS. For example, the following FILEDEF commands define new data sets to contain the session script and the session output:

```
FILEDEF FSC3270I DISK FSSCRIPT IN A (DISP MOD
FILEDEF FSC3270O DISK FSC3270 OUTBASE A (DISP MOD
```

You must invoke FOCUS without a profile:

```
EX FOCUS (NOPROF
```

If you need a profile, execute the profile as the first command after invoking FOCUS.

The following FOCUS session issues a TABLE request against the MOVIES data source. This session is recorded by FOCREPLAY as a result of executing the REXX EXEC containing the FILEDEF commands for DDNAMEs FSC3270I and FSC3270O.

Issue the following TABLE request:

```
>  > TABLE FILE MOVIES
> PRINT RATING DIRECTOR TITLE/A20
> BY CATEGORY
> WHERE CATEGORY EQ 'DRAMA' OR 'MYSTERY' OR 'CHILDREN'
> END

 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press Enter to display the first screen of the report:

```
PAGE     1


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
CHILDREN  NR                         SMURFS, THE
          G       BARTON C.          SHAGGY DOG, THE
          NR                         SCOOBY-DOO-A DOG IN
          G       GEROMINI           ALICE IN WONDERLAND
          NR                         SESAME STREET-BEDTIM
          NR                         ROMPER ROOM-ASK MISS
          NR      DISNEY W.          SLEEPING BEAUTY
          G       DISNEY W.          BAMBI
DRAMA     R       LUMET S.           DOG DAY AFTERNOON
MYSTERY   PG      HITCHCOCK A.       REAR WINDOW
          PG      HITCHCOCK A.       VERTIGO
          R       GRANT M.           FATAL ATTRACTION
          NR      HITCHCOCK A.       NORTH BY NORTHWEST
          R       CRONENBERG D.      DEAD RINGERS
          R       LUMET S.           MORNING AFTER, THE
          R       HITCHCOCK A.       PSYCHO
          PG13    HITCHCOCK A.       BIRDS, THE

                                                        MORE
```

Press Enter to display the second screen of the report:

```
PAGE     2


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
MYSTERY   R       BECKER H.          SEA OF LOVE
```

```
                                END OF REPORT
```

Issue the FIN command to exit FOCUS:

```
FIN
```

Once you issue the FIN command, you are no longer in FOCUS, so recording stops. However, if you want to invoke FOCUS at this point without recording the new session, you must free the FOCREPLAY FILEDEFs. If you do not, the new session's script will be appended to the existing script file and its output will be appended to the existing output file.

Example: Recording a Session Under z/OS

You can create the two files needed for FOCREPLAY input mode prior to executing the CLIST used to invoke FOCUS or in the CLIST. If you create these files in the CLIST, you must then free the allocations and reallocate them with DISP MOD. For example, the following commands allocate new data sets to contain the session script and the session output:

```
ALLOC F(FSC3270I) NEW DA('USER1.FSC.SCRIPT') ,
                  SPACE(50,10) TRACKS LRECL(23472) BLKSIZE(23476) ,
                  DSORG(PS) RECFM(V)
ALLOC F(FSC3270O) NEW DA('USER1.FSC.OUTBASE') ,
                  SPACE(50,10) TRACKS LRECL(23472) BLKSIZE(23476) ,
                  DSORG(PS) RECFM(V)"
```

The following commands free these allocations and reallocate the data sets with DISP MOD:

```
FREE F(FSC3270I)
FREE F(FSC3270O)
ALLOC F(FSC3270I) REUSE MOD DA('USER1.FSC.SCRIPT')
ALLOC F(FSC3270O) REUSE MOD DA('USER1.FSC.OUTBASE')
```

The following FOCUS session issues a TABLE request against the MOVIES data source. This session is recorded by FOCREPLAY as a result of executing the CLIST containing the allocations for DDNAMEs FSC3270I and FSC3270O.

Issue the following TABLE request:

```
>  > TABLE FILE MOVIES
> PRINT RATING DIRECTOR TITLE/A20
> BY CATEGORY
> WHERE CATEGORY EQ 'DRAMA' OR 'MYSTERY' OR 'CHILDREN'
> END

 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press Enter to display the first screen of the report:

```
PAGE     1


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
CHILDREN  NR                         SMURFS, THE
          G       BARTON C.          SHAGGY DOG, THE
          NR                         SCOOBY-DOO-A DOG IN
          G       GEROMINI           ALICE IN WONDERLAND
          NR                         SESAME STREET-BEDTIM
          NR                         ROMPER ROOM-ASK MISS
          NR      DISNEY W.          SLEEPING BEAUTY
          G       DISNEY W.          BAMBI
DRAMA     R       LUMET S.           DOG DAY AFTERNOON
MYSTERY   PG      HITCHCOCK A.       REAR WINDOW
          PG      HITCHCOCK A.       VERTIGO
          R       GRANT M.           FATAL ATTRACTION
          NR      HITCHCOCK A.       NORTH BY NORTHWEST
          R       CRONENBERG D.      DEAD RINGERS
          R       LUMET S.           MORNING AFTER, THE
          R       HITCHCOCK A.       PSYCHO
          PG13    HITCHCOCK A.       BIRDS, THE

                                                        MORE
```

Press Enter to display the second screen of the report:

```
PAGE     2


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
MYSTERY   R       BECKER H.          SEA OF LOVE
```

```
                                END OF REPORT
```

Issue the FIN command to exit FOCUS:

```
FIN
```

Once you issue the FIN command, you are no longer in FOCUS, so recording stops. However, if you want to invoke FOCUS at this point without recording the new session, you must free the FOCREPLAY allocations. If you do not, the new session's script will be appended to the existing script file and its output will be appended to the existing output file.

Replaying a Recorded FOCUS Session

How to:

Replay a Recorded FOCUS Session Interactively

Reference:

FOCREPLAY PFKeys

Example:

Replaying a Recorded FOCUS Session Interactively Under z/VM

Replaying a Recorded FOCUS Session Interactively Under z/OS

You can replay a recorded FOCUS session interactively or in a batch job. **Note:** Do *not* allocate DDNAMEs SYSIN or SYSPRINT in the FOCREPLAY batch job.

If you replay the session interactively, you must press a key in order to proceed from one line to the next in the script. Press Enter to move through the script line by line or, if your FOCUS session is accepting input, you can press a PFkey to move more quickly through the script.

Procedure: How to Replay a Recorded FOCUS Session Interactively

1. Allocate or FILEDEF the required DDNAMES in your CLIST that invokes FOCUS. Allocate:
   - ❏ DDNAME FSC3270Q to the file containing the previously recorded script.
   - ❏ DDNAME FSC3270O to the file to receive the replayed session output.
   - ❏ DDNAME FSC3270I as DUMMY
2. Execute your REXX EXEC or CLIST that invokes FOCUS.

   The first line in the script displays on the screen.
3. Press Enter to invoke the next line in the script or, if your FOCUS session is accepting input, you can press a PFkey to move more quickly through the script.
4. Continue pressing Enter or a PFkey to move through the script.

   When the FIN command is replayed, the FOCUS session ends, and the replay stops.

   Note: If you want to invoke FOCUS at this point without replaying the session again, you must clear the FOCREPLAY FILEDEFs or free the FOCREPLAY allocations. If you do not, the new session output will be appended to the existing output file.

Reference: FOCREPLAY PFKeys

If your FOCUS session is accepting input, you can press the Enter key to step through the session line by line, or you can press one of the following PFkeys:

PFkey	Action
F1	Plays the next 10 script lines with no confirmation.
F2	Plays the next 20 script lines with no confirmation.
F3	Plays the next 30 script lines with no confirmation.
F4	Plays the next 40 script lines with no confirmation.
F5	Plays the next 50 script lines with no confirmation.
F6	Plays the next 60 script lines with no confirmation.
F7	Plays the next 70 script lines with no confirmation.
F8	Plays the next 80 script lines with no confirmation.
F9	Plays the next 90 script lines with no confirmation.
F10	Plays the next 100 script lines with no confirmation.
F11	Plays to the end of the script or to a break point with no confirmations.
F12	Quit (exits FOCUS by issuing Abend U610-1). At this point you must log off and on again to clean up after the abend.
F24	Exits to a live FOCUS session. If you are re-recording a session (by allocating an input file), you can input new key strokes at this point.

Example: Replaying a Recorded FOCUS Session Interactively Under z/VM

The REXX EXEC that invokes FOCUS has the following FOCREPLAY FILEDEFs:

- The previously recorded script:

```
'FILEDEF FSC3270Q DISK FSSCRIPT IN A'
```

- The output of the replayed session:

```
'FILEDEF FSC3270O DISK FSC3270 OUTNEW A (DISP MOD'
```

- The dummy FILEDEF for DDNAME FSC3270I:

```
'FILEDEF FSC3270I DUMMY'
```

Once the REXX EXEC executes and invokes FOCUS, the first line of the script file displays:

```
>  >  TABLE FILE MOVIES
```

You can press Enter, or you can press one of the FOCREPLAY PFkeys described in *FOCREPLAY PFKeys*.

For this example, press Enter to execute this line. The next line displays:

```
>  PRINT RATING DIRECTOR TITLE/A20
```

Press Enter to execute this line. The next line displays:

```
>  BY CATEGORY
```

Press Enter to execute this line. The next line displays:

```
>  WHERE CATEGORY EQ 'DRAMA' OR 'MYSTERY' OR 'CHILDREN'
```

Press Enter to execute this line. The next line displays:

```
 >  END
```

Press Enter to execute this line. The report summary lines display:

```
NUMBER OF RECORDS IN TABLE=       18  LINES=     18

PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press Enter. The first screen of report output displays:

```
PAGE     1


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
CHILDREN  NR                         SMURFS, THE
          G       BARTON C.          SHAGGY DOG, THE
          NR                         SCOOBY-DOO-A DOG IN
          G       GEROMINI           ALICE IN WONDERLAND
          NR                         SESAME STREET-BEDTIM
          NR                         ROMPER ROOM-ASK MISS
          NR      DISNEY W.          SLEEPING BEAUTY
          G       DISNEY W.          BAMBI
DRAMA     R       LUMET S.           DOG DAY AFTERNOON
MYSTERY   PG      HITCHCOCK A.       REAR WINDOW
          PG      HITCHCOCK A.       VERTIGO
          R       GRANT M.           FATAL ATTRACTION
          NR      HITCHCOCK A.       NORTH BY NORTHWEST
          R       CRONENBERG D.      DEAD RINGERS
          R       LUMET S.           MORNING AFTER, THE
          R       HITCHCOCK A.       PSYCHO
          PG13    HITCHCOCK A.       BIRDS, THE

                                                          MORE
```

Press Enter. The next screen of report output displays:

```
PAGE     2

CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
MYSTERY   R       BECKER H.          SEA OF LOVE




                                  END OF REPORT
```

Press Enter. The next line of the script displays:

```
>  >  FIN
```

Press Enter to execute this line.

Once the FIN command executes, you exit FOCUS, so replay ends. However, if you want to invoke FOCUS at this point without replaying the session again, you must clear the FOCREPLAY FILEDEFs. If you do not, the new session output will be appended to the existing output file.

After replaying a session, you have one output file from the recorded session and another output file from the replayed session. You can compare these two output files.

Example: Replaying a Recorded FOCUS Session Interactively Under z/OS

The CLIST that invokes FOCUS has the following FOCREPLAY allocations:

- The previously recorded script:

  ```
  ALLOC F(FSC3270Q) DA ('USER1.FSC.SCRIPT') MOD
  ```

- The output of the replayed session:

  ```
  ALLOC F(FSC3270O) DA ('USER1.FSC.OUTNEW') MOD
  ```

- The dummy allocation for DDNAME FSC3270I:

```
ALLOC F(FSC3270I) DUMMY
```

Once the CLIST executes, the first line of the script file displays:

```
>  >  TABLE FILE MOVIES
```

You can press Enter, or you can press one of the FOCREPLAY PFkeys described in *FOCREPLAY PFKeys*.

For this example, press Enter to execute this line. The next line displays:

```
>  PRINT RATING DIRECTOR TITLE/A20
```

Press Enter to execute this line. The next line displays:

```
>  BY CATEGORY
```

Press Enter to execute this line. The next line displays:

```
>  WHERE CATEGORY EQ 'DRAMA' OR 'MYSTERY' OR 'CHILDREN'
```

Press Enter to execute this line. The next line displays:

```
 >  END
```

Press Enter to execute this line. The report summary lines display:

```
NUMBER OF RECORDS IN TABLE=       18  LINES=     18

PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press Enter. The first screen of report output displays:

```
PAGE     1


CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
CHILDREN  NR                         SMURFS, THE
          G       BARTON C.          SHAGGY DOG, THE
          NR                         SCOOBY-DOO-A DOG IN
          G       GEROMINI           ALICE IN WONDERLAND
          NR                         SESAME STREET-BEDTIM
          NR                         ROMPER ROOM-ASK MISS
          NR      DISNEY W.          SLEEPING BEAUTY
          G       DISNEY W.          BAMBI
DRAMA     R       LUMET S.           DOG DAY AFTERNOON
MYSTERY   PG      HITCHCOCK A.       REAR WINDOW
          PG      HITCHCOCK A.       VERTIGO
          R       GRANT M.           FATAL ATTRACTION
          NR      HITCHCOCK A.       NORTH BY NORTHWEST
          R       CRONENBERG D.      DEAD RINGERS
          R       LUMET S.           MORNING AFTER, THE
          R       HITCHCOCK A.       PSYCHO
          PG13    HITCHCOCK A.       BIRDS, THE

                                                          MORE
```

Press Enter. The next screen of report output displays:

```
PAGE     2

CATEGORY  RATING  DIRECTOR           TITLE
--------  ------  --------           -----
MYSTERY   R       BECKER H.          SEA OF LOVE

                                  END OF REPORT
```

Press Enter. The next line of the script displays:

```
>  >  FIN
```

Press Enter to execute this line.

Once the FIN command executes, you exit FOCUS, so replay ends. However, if you want to invoke FOCUS at this point without replaying the session again, you must free the FOCREPLAY allocations. If you do not, the new session output will be appended to the existing output file.

After replaying a session, you have one output file from the recorded session and another output file from the replayed session. You can compare these two output files.

Comparing a Recorded Session With a Replayed Session

Example:

Comparing Two Sessions Using the XCOMPARE Facility on z/VM

Comparing Two Sessions Using the ISPF SuperCE Utility

FOCREPLAY does not compare output files. You can use any file comparison tool installed at your site to perform the comparison. When you view the results, you must determine which differences are acceptable and do not indicate a change in your application's operation.

Example: Comparing Two Sessions Using the XCOMPARE Facility on z/VM

Issue the following command at the Ready prompt:

```
XCOMPARE FSC3270 OUTBASE A FSC3270 OUTNEW A (DISK
```

The result of the comparison is placed in the file XCOMPARE LISTING A1:

```
1COMPARING 'FSC3270  OUTBASE  A1' WITH 'FSC3270  OUTNEW   A1'
0FILE 1      1  O: FOCUS  7.3.7   06/06/2006  16.06.11  GEN01.01
 FILE 2      1  O: FOCUS  7.3.7   06/06/2006  16.18.02  GEN01.01
0FILE 1     36  I00007: {Enter}{024-002}
0FILE 2     60  I00007: {Enter}{024-002}
0FILE 1     68  I00008: {Enter}{024-002}
0FILE 2     92  I00008: {Enter}{024-002}
0COMPARISON COMPLETED - RETURN CODE IS     8.
```

The differences in the two files are the lines with the date and time stamp and lines inserted by FOCREPLAY. These differences are acceptable and do not indicate a change in the operation of the application.

Example: Comparing Two Sessions Using the ISPF SuperCE Utility

Go to option 3.13 (the SuperCE Utility) from the ISPF Primary Option Menu.

Enter the names of the two output files on the screen and press Enter to start the comparison. This example accepts the default comparison options:

```
sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
                              SuperCE Utility
Command ===>

New DS Name  . . . 'USER1.FSC.OUTBASE'
Old DS Name  . . . 'USER1.FSC.OUTNEW'
PDS Member List              (blank/pattern - member list, * - compare all)
   (Leave New/Old DSN "blank" for concatenated-uncataloged-password panel)
   Compare Type               Listing Type               Display Output
   2  1. File                 2  1. OVSUM                1  1. Yes
      2. Line                    2. Delta                   2. No
      3. Word                    3. CHNG                    3. Cond
      4. Byte                    4. Long                    4. UPD

Listing DSN  . . . . SUPERC.LIST
```

The results are placed in a file named USER1.SUPERC.LIST and are automatically displayed when the comparison is complete. In this example, the only lines that are different are those that show the date and time of the two runs. This difference does not indicate a change in the operation of the application:

```
                    LISTING OUTPUT SECTION (LINE COMPARE)

 ID        SOURCE LINES

 I - O:    FOCUS   7.3.7    06/07/2006   15.46.14
 D - O:    FOCUS   7.3.7    06/07/2006   15.48.43


                      LINE COMPARE SUMMARY AND STATISTICS

  106 NUMBER OF LINE MATCHES                1   TOTAL CHANGES (PAIRED+NONPAIRED CHNG)
    0 REFORMATTED LINES                     1   PAIRED CHANGES (REFM+PAIRED INS/DEL)
    1 NEW FILE LINE INSERTIONS              0   NON-PAIRED INSERTS
    1 OLD FILE LINE DELETIONS               0   NON-PAIRED DELETES
  107 NEW FILE LINES PROCESSED
  107 OLD FILE LINES PROCESSED

LISTING-TYPE = DELTA      COMPARE-COLUMNS =    1:23468     LONGEST-LINE = 126
PROCESS OPTIONS USED: NONE

 ISRS004I LISTING LINES MAY BE TRUNCATED DUE TO LIMITING OUTPUT LINE WIDTH.
```

Stopping Replay at a Break Point

If you want to stop the replay at a break point, edit the input file and add the following on a line by itself right in front of the last line you want executed before the break point:

```
<BREAK>
```

On replay (online only), the input sequence is replayed up to the line following the break point, and then the keyboard is opened for stepwise motion through the remainder of the script using existing PF key functions. This technique enables you get to a specific point in a long script very quickly.

If you use F10 or F11 to move down in the script, and a break is encountered along the way, replay stops at the break point rather than proceeding to where the PF key would have taken you.

You can set multiple break points in a script file by inserting multiple <BREAK> commands in the file.

Re-recording From the Middle or End of a Script

In order to continue recording a partially recorded session, you must allocate DDNAME FSC3270I to a file that will contain the new version of the script. The existing script must be in a different file, allocated to DDNAME FSC3270Q. In addition, if you allocate an existing output file to DDNAME FSC3270O (to receive the session output), the new output will appended to the existing output. If you want the re-recorded session to go to a separate file, allocate a new output file to DDNAME FSC3270O.

To start recording where you left off last time, remove the FIN command and execute replay mode online. The existing script will be re-recorded to the new script file. At the end of the existing script, the keyboard will open for input, and your key strokes will be recorded at the end of the new script file.

To start recording in the middle of a script file, edit the existing script file and add the following on a line by itself at the point in the script at which you want to start re-recording:

```
<RECORD>
```

In an online session, the replay tool plays the existing script and re-records it to the new script file. When it encounters the <RECORD> command, it opens the keyboard for further input that will be recorded and added to the end of the new script file.

This technique eliminates trailing lines after a mistake and enables you to re-record the remainder of the session.

In this way, you do not have to start recording from scratch when a mistake is made during the recording session or when you stop at some point but want to continue recording later.

You must allocate the new script file to DDNAME FSC3270Q when replaying the corrected or extended session.

8 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).

Topics:

- 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

In this section:

How to:

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 280 and *Distributed Execution* on page 288), 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.

In z/VM, you issue a FILEDEF to name the configuration file. In z/OS, you allocate the ddname for the file, containing configuration information, to either a sequential file or a member of a partitioned data set.

DNS Names Support

How to:

Invoke DNS Names Support

Example:

Using 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

In this section:

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

How to:

Example:

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/VM or z/OS user ID) are used to sign on onto the server.

Sending Requests to a Remote Server

How to:

Execute a Request Remotely Using the REMOTE EX Command

Example:

Executing a Request Remotely Using the REMOTE EX Command

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

How to:

Execute Remote Procedures in Remote Execution Mode

Example:

Executing Requests Using -REMOTE BEGIN and -REMOTE END

Executing a Remote Procedure in Remote Execution Mode

Using -INCLUDE to Call a FOCEXEC Stored on the Client

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

How to:

Display 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

How to:

Terminate a Remote Session With the REMOTE FIN Command

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

How to:

Query Remote Session Parameter Settings

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

In this section:

How Location Transparency Works

Logging On to the Server With Distributed Execution

Joining Data Sources Across Platforms With Distributed Execution

Issuing SQL Commands to the Server With Distributed Execution

Executing Stored Procedures With Distributed Execution

Using SQL Passthru With Distributed Execution

How to:

Implement Distributed Execution

Example:

Storing a Server Name in an Access File

Submitting a Request Using SUFFIX=EDA

Using a Remote Multi-Segment Master and Access File

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.

On z/VM, you specify an Access File name that matches your Master File name, with a file type of FOCSQL. For example:

```
CAR MASTER A - Master File
CAR FOCSQL A - Access File
```

On z/OS, your Access File member names must also 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

How to:

Set the Server Destination With Distributed Execution

Send a User ID and Password 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

How to:

Execute a Stored Procedure Using Distributed Execution

Query Server Data Adapter Settings

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.

9 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.

Topics:

- ❏ Overview of FOCLOG
- ❏ Implementing FOCLOG
- ❏ Information Captured in the FOCLOG File
- ❏ FOCLOG Reporting

Overview of FOCLOG

In this section:

How Logging is Implemented

The Log Data Set

Sample FOCLOG Configuration Scenarios

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, FOCUS detects the presence of member FOCUSLOG in any of the data sets allocated to DDNAME ERRORS on z/OS or the FOCUSLOG ERRORS file on z/VM. 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 delete or rename the reference to the log file within the FOCUSLOG file.

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.

On z/OS, 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.

On z/VM, your SFS (Shared File System) Administrator can see the log full condition recorded in the Filepool Console log.

The z/OS ISPF or z/VM FILELIST utility can be used to examine the date and time of the last update of the log.

The Log Data Set

On z/OS, 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.

On z/VM, the first record of the FOCUSLOG ERRORS file contains the fully qualified name of the SFS directory.

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:

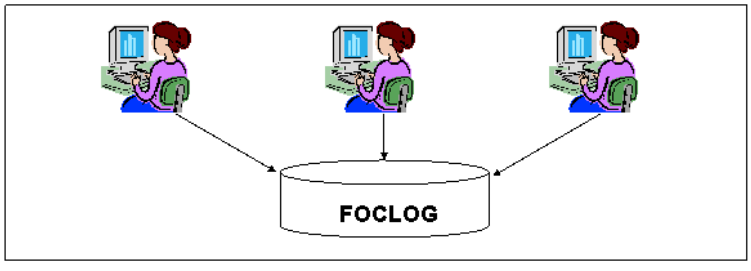
Type of record	Size in bytes	Number of Occurrences
Session	168	One per FOCUS session or batch job
Command	129 or 173	One per FOCUS command that accesses data
File	28 or 116	One per MASTER file referenced by the command
Dataset	68	One per data set still allocated at the end of the job

The detailed description of each record is in the distributed FOCLOG Master File and in *Information Captured in the FOCLOG File* on page<$pagenum>. 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

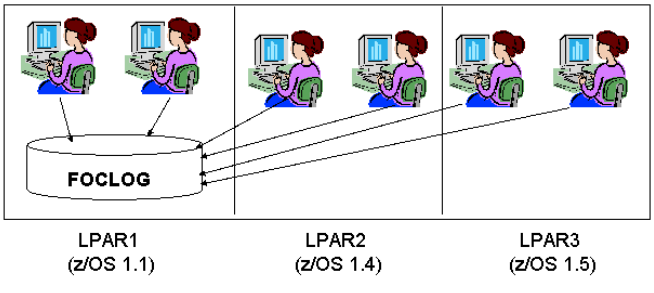
In the following diagram of a single LPAR configuration, one FOCLOG file records the usage of all FOCUS users in that LPAR:

Single LPAR Configuration



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.

Multiple LPAR Configuration
in a Sysplex Environment



Implementing FOCLOG

In this section:

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 create a log file and activate it by specifying the fully qualified name of the SFS directory in the FOCUSLOG file.

If your site standards require you to implement new features in a test environment, create a temporary FOCUSLOG ERRORS file.

- ❏ On z/OS, create a copy of your FOCUS production CLIST and concatenate your ERRORS library in front of the FOCUS production ERRORS library.
- ❏ On z/VM, place your FOCUSLOG ERRORS file on a private minidisk that is higher in the search order than the FOCUS production minidisk.

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

Reference:

Usage Notes for FOCLOG

These steps provide a high level overview of the FOCLOG implementation process:

1. Allocate the FOCLOG data set.

2. Activate FOCLOG.
3. Validate the FOCLOG configuration.
4. Move FOCLOG to your production FOCUS environment.
5. 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

How to:

Allocate the FOCLOG Log File on z/OS

Estimate the Size of the 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 303.

Procedure: How to Allocate the FOCLOG Log File on z/OS

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 FOCLOGSU.
- 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 numbers 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 don't 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 day's end, 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

Example:

Requesting a Default Form of the Log on z/OS

Requesting a Default Form of the Log on z/VM

Requesting a Detailed Form of the Log on z/OS

Requesting a Detailed Form of the Log on z/VM

Requesting a Session Summary Form of the Log on z/OS

Requesting a Session Summary Form of the Log on z/VM

Reference:

Notes on Activation and Deactivation of Logging

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 three components:

1. On z/OS, the presence of a FOCUSLOG member in the concatenation of data sets allocated to DDNAME ERRORS. On z/VM, the existence of a FOCUSLOG ERRORS file on a disk accessed by FOCUS users.
2. The FOCUSLOG file must contain the name of the FOCLOG log file.
3. An indicator for the type of log desired.

Example: Requesting a Default Form of the Log on z/OS

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, Access Files, and Master Files.

```
FLHLQ.FOCLOG.DATA
```

Example: Requesting a Default Form of the Log on z/VM

The FOCUSLOG file in this example points to an SFS directory named VMSYSU:FOCLOG.DATA. Because the directory 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, Access Files, and Master Files:

```
VMSYSU:FOCLOG.DATA
```

Example: Requesting a Detailed Form of the Log on z/OS

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 Detailed Form of the Log on z/VM

The FOCUSLOG file in this example points to an SFS directory named VMSYSU:FOCLOG.DATA. Because the directory name is followed by the keyword DETAIL, this generates 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:

```
VMSYSU:FOCLOG.DATA DETAIL
```

Example: Requesting a Session Summary Form of the Log on z/OS

This FOCUSLOG member points to the same log file as the one defined in *Requesting a Default Form of the Log on z/OS* on page<$pagenum>, 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 SUMMARY
```

Example: Requesting a Session Summary Form of the Log on z/VM

The FOCUSLOG file in this example points to an SFS directory named VMSYSU:FOCLOG.DATA. Because the directory name is followed by the keyword SUMMARY, this generates a session summary form of the log containing only information contained in the session segment of the FOCLOG Master File.

```
VMSYSU:FOCLOG.DATA SUMMARY
```

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 contains the name of a log file, 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. Therefore, FOCLOG does not produce any unnecessary overhead or use additional CPU cycles for capturing usage data.

Because logging depends on having a FOCUSLOG file that points to a log file, you can deactivate FOCLOG by taking any of the following actions. These actions are listed in the recommended order:

- Rename the log file itself so that the FOCUSLOG file no longer points to a valid log file.

  You can also 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. **Note:** Log replacement should be done as quickly as possible to avoid losing data that should be captured in the log.

- Rename the pointer to the log file in your FOCUSLOG member. For example, assume the FOCUSLOG file contains the following log file name:

  On z/OS:

  ```
  FLHLQ.FOCLOG.DATA
  ```

  On z/VM:

  ```
  VMSYSU:FOCLOG.DATA
  ```

  Change this entry in FOCUSLOG so that it no longer points to a valid log file or SFS directory. For example, if no file exists with the name FOCLOG.DATAOFF, change the pointer in FOCUSLOG to reference this non-existent file:

  On z/OS:

  ```
  FLHLQ.FOCLOG.DATAOFF
  ```

  On z/VM:

  ```
  VMSYSU:FOCLOG.DATAOFF
  ```

- Rename the FOCUSLOG file so that it is not found under the name FOCUSLOG. For example, change the name to FOCLGOFF.

Validating the FOCLOG Configuration on z/OS

Example:

Sample FOCUS Session on z/OS

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, please 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, please 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 on z/OS

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
******* VM:   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
******* VM:   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.

Validating the FOCLOG Configuration on z/VM

Example:

Sample FOCUS Session on z/VM

This step validates your FOCLOG configuration and enables you to confirm that the log file was accessed 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 the EXEC that your customers use to access FOCUS. This EXEC should FILEDEF the production versions of the ERRORS, MASTER, and FOCEXEC files.
2. Run the test EXEC to enter FOCUS.
3. Next, execute a request to populate the log:

   Edit the FLVALPOP FOCEXEC and delete the following lines near the top of the file, then save the edited file:

   ```
   00006 -? TSO DDNAME CAR
   00007 -IF &DSNAME EQ ' ' THEN GOTO ALLOC1
   00008 DYNAM FREE DD CAR
   00009 -ALLOC1
   00010 DYNAM ALLOC DD CAR  RECFM F LRECL 4096 BLKSIZE 4096 -
   00011       DSORG PS                                      -
   00012       SPACE 1,3 CYL
   00013 -RUN
   ```

   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     FOCUS   A1      ON 09/15/2007 AT 09.06.57
    CAR     FOCUS   A1      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, please 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 EXEC to enter FOCUS again.

6. You will now execute a request to produce a validation report from the log.

   a. Issue the following commands to ACCESS and FILEDEF your log file so that you can issue a report request against it. (**Note:** Before pressing Enter, make sure the data set name is the name of the log file you specified in the FOCUSLOG file.). For example:

```
ACCESS VMSYSU:USER1.FOCLOG B
CMS FILEDEF FOCLOG DISK FOCLOG DATA B
```

   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. **Note:** If you have problems running this request, please 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 XEDIT and delete the lines that contain the data.

Example: Sample FOCUS Session on z/VM

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   11/02/2007  09.50.12  GEN231
OBSERVED CPU:
****** Processor 00AF5A Model 2066-000 Max 001 >Site GEN2
LICENSED CPU(S):
****** Processor 005394 Model 9672-0F0 Max 001 >Registration 57E3C86A0887
```

The following command executes the request that populates the log:

```
>  > ex flvalpop
```

The following messages display:

```
NEW FILE CAR     FOCUS   A1 ON 11/02/2007 AT 09.55.09
 CARFL   FOCUS   A1 ON 11/02/2007 AT 09.55.11
 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 ACCESS and FILEDEF the log file:

```
ACCESS VMSYSU:USER1.FOCLOG B
CMS FILEDEF FOCLOG DISK FOCLOG DATA B
```

Then, we go back into FOCUS:

```
FOCUS  7.6.3   11/02/2007  09.55.10  GEN231
OBSERVED CPU:
****** Processor 00AF5A Model 2066-000 Max 001 >Site GEN2
LICENSED CPU(S):
****** Processor 005394 Model 9672-0F0 Max 001 >Registration 57E3C86A0887
```

Now we issue the FLVALRPT request to view information 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
```

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 XEDIT) before putting the log into production.

Running the FOCLOG Reports

FOCLOG is now configured. See *FOCLOG Reporting* on page<$pagenum>, for instructions about reporting on the captured data.

Information Captured in the FOCLOG File

Reference:

Session Segment Information

Command Segment Information

Master Segment Information

Capturing the Access File Data Set Name

Data Set Segment Information

Master File Structure Diagram

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's session.
- The command segment provides information about FOCUS commands issued during a user's 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's 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)

Field Name	Description
OPSYS	Operating system (format A8)
OPSYSREL	OS version/release (format A8)
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)
MODEL_ID	Processor Model ID (format A8)
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)
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 (e.g. TABLE, MODIFY) (format A8)

Field Name	Description
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)
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)
OUTPUT_TYPE	DEFINE: output type name (format A8). Decode OUTFLAG table

Field Name	Description
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: IF FOCEXEC LT 'A' THEN ' ', ELSE FOCEXEC (format A8)

Reference: Master Segment Information

Field Name	Description
RECTYPE	Master segment record type (MASS) (format A4)
FNAME	Master File name (format A8)
SUFF	File suffix (FOC, FIX, etc.) (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<$pagenum>.
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)

Field Name	Description
DDEXCP	EXCPs for ddname (format A7)
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

In this section:

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

How to:

Reference:

Example:

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 or FILEDEF 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.';
  ```

- If you are running on z/VM, replace VMSYSU:FOCLOG.DATA with your SFS directory name in the following line:

  ```
  CMS FILEDEF FOCLOG DISK VMSYSU:FOCLOG.DATA
  ```

- If you are running on z/OS, 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 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 327.

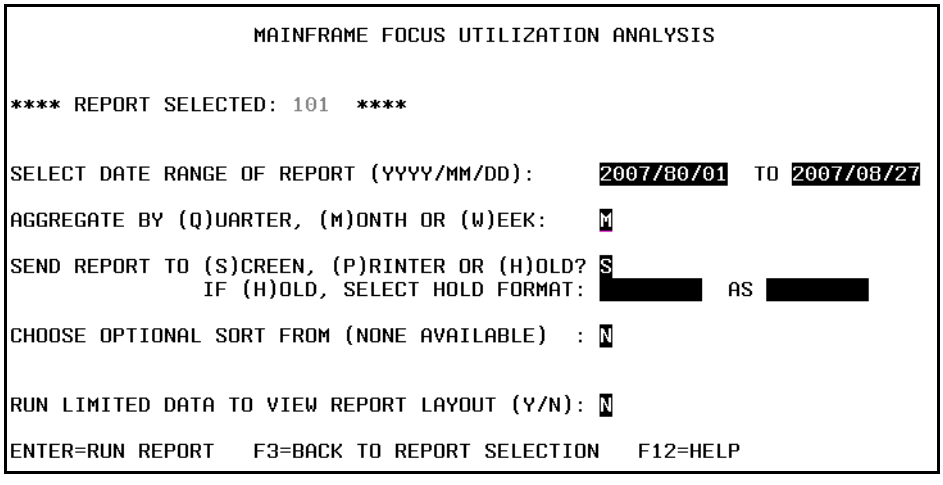
To select a specific report, enter its three-digit report number. If you know a report's three-digit number, you can enter that number from any screen.

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:



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 FLRPT204.

First, make sure you have access to the FOCLOG log file and that you added the appropriate DYNAM or FILEDEF 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 to open the FILE screen:

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

DIMENSION:  USER    FILE    PROCEDURE    USAGE    CUSTOM    ADHOC
                    ****

      201 - POSSIBLE EXTRACT FILES BEING USED
      202 - FILES ORIGINATING LARGE HOLD FILE EXTRACTS
      203 - FILES USED BY USERS
      204 - BATCH AND ONLINE FILE USAGE
      205 - REPORTS AND TRANSACTIONS AGAINST DATA SOURCES
      206 - FILES USING THE MOST CPU ON REPORTS
      207 - DATA SOURCE  TYPES ACCESSED BY FOCUS
      208 - MOST RECORDS EXTRACTED DURING ONLINE SESSIONS


      SELECT: ___

 F3=EXIT    F7=PREVIOUS SCREEN    F8=MORE REPORTS    F12=HELP
```

Enter the number 204 in the SELECT field to run FLRPT204. The Options screen opens:

```
                    MAINFRAME FOCUS UTILIZATION ANALYSIS

**** REPORT SELECTED: 204  ****

SELECT DATE RANGE OF REPORT (YYYY/MM/DD):      2007/07/01  TO 2007/10/26

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   (D)URATION (C)PU (E)XCP :

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.
               JULY  1, 2007 -      OCTOBER 26, 2007
                     BATCH AND ONLINE FILE USAGE


BATCH/ SESSION                DURATION    CPU USAGE
ONLINE  COUNT  FILE TYPE     HH:MM:SS     HH:MM:SS  EXCPS/1000
------ ------- --------- ----------- -----------  ----------
BATCH        2                 07:28       00:05           0
               FOCUS         1:58:47       01:06           0

TOTAL                        2:06:16       01:11           0
-------------------------------------------------------------
ONLINE       8                 04:14       00:07           0
               DB2               00:00       00:00           0
               FIXED          2:29:16       00:34           0
               FOCUS         10:33:08       06:30           0
               SQL/DS         1:09:58       00:04           0


  PAGE    1              2007/10/26 13.27.09     FLRPT204
```

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, please 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

Reference:

User Reports - 100 Series

File Dimension Reports - 200 Series

Procedure Dimension Reports - 300 Series

Usage Dimension Reports - 400 Series

Each of the following tables describes the reports available for a specific dimension. Each report has an internal name of the form:

FLRPT*nnn*

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 during the period. The report is sorted by CPU usage for Batch jobs and Online sessions.	High CPU 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 via 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 column tracks 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 userid 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 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<$pagenum> for a description of the FOCLOG Master File in order to develop your own schemes for collapsing user IDs into local groups.
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 or number of runs, and possible imbalances between reports and update operations.

No.	Report Title and Contents	Interpreting Report Output
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 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 and CPU 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.

No.	Report Title and Contents	Interpreting Report Output
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 filename 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.
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 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 resort 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 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 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, 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 savings. This report has the potential of identifying procedures that can experience large CPU 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 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	What to look for
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.
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.

No.	Report name and contents	What to look for
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-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.
405	**Hourly CPU Usage Accumulated (Graph)** bar graph accumulates CPU 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 usage on a particular day.

No.	Report name and contents	What to look for
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, etc.). 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 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 usage is inline with industry trends.
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.

10 Storing Terminal Lines in Memory: The Session Monitor

The Session Monitor facility enables you to review and save the input and output you generate during online FOCUS sessions.

Topics:

Session Monitor Overview

How to:

Activate or Deactivate the Session Monitor

Using the Session Monitor, you can review what you have done, review error messages, save reports displayed online, and save FOCUS commands as FOCEXECs for later execution.

As you use FOCUS, the Session Monitor stores your terminal input and output in a stack in memory, appending new lines to the bottom of the stack. The stack can hold about 100 lines (8000 bytes). After this limit is reached, old lines are deleted from the top of the stack as new lines are added at the bottom. Thus, the stack stores the most recent 8000 bytes of your session.

Input from FOCEXECs and output from special full-screen facilities, such as FIDEL, HotScreen, and TableTalk, and from the TRACE and ECHO options of MODIFY requests, are not appended to the stack. To learn how to record output from the TRACE and ECHO options, see the *Maintaining Databases* manual. To append report output to the stack, you can issue the SET SCREEN = OFF command to prevent the output from being displayed using HotScreen. To append FOCEXEC lines to the stack, execute the FOCEXEC with ECHO=ON or ECHO=ALL.

Syntax: How to Activate or Deactivate the Session Monitor

```
SET SM = {OFF|ON|INPUT}
```

where:

OFF

Turns off the Session Monitor and deletes the stack from memory. OFF is the default value.

ON

Turns on the Session Monitor. The Session Monitor begins stacking your terminal input and output in memory until you end your FOCUS session or you turn off the Session Monitor.

INPUT

Turns on the Session Monitor, but the Session Monitor stacks only input lines, not output.

You can inquire about the current status of the Session Monitor by issuing the ? SET SM query.

Displaying the Session Monitor Stack

Reference:

Session Monitor Line Categories

Session Monitor Display Commands

Example:

Displaying Lines From the Session Monitor Stack

The Session Monitor has a number of commands for displaying the contents of the stack. These commands move an internal pointer forward and backward and display the contents of the stack starting at the location of the pointer (the Session Monitor commands themselves are not appended to the stack).

Initially, the pointer resides at the bottom of the stack (after the most recent entry). You can always reset the pointer at the bottom by pressing Enter.

Reference: Session Monitor Line Categories

Session Monitor categorizes terminal lines as one or more of the following:

- **Input lines.** These consist of any lines you type at the FOCUS command prompt, including commands, responses to prompts, or corrections in response to an error message. It can also include FOCEXEC lines if ECHO = ON or ALL.
- **Output lines.** These consist of the lines displayed by FOCUS, including messages, report output if SCREEN = OFF, and FOCEXEC lines if ECHO = ON or ALL.
- **Command lines.** These consist of FOCUS commands and subcommands.

Reference: Session Monitor Display Commands

You can enter Session Monitor commands at any time during a FOCUS session, except when you are using special full-screen facilities, such as FIDEL, Hot Screen, or TableTalk. You can even enter Session Monitor commands while being prompted for data or in the middle of entering other FOCUS commands, as in this TABLE request:

```
TABLE FILE CAR
PRINT CAR
/PC
```

If you enter a Session Monitor command during a data prompt, press Enter to return to the prompt.

You can include Session Monitor commands in FOCEXECs.

The following table lists the Session Monitor display commands:

Command	Command Name	**Description**
/PP	Previous Page	Moves the pointer back one page (20 lines) and displays the page. Repeating the command scrolls the pointer backward toward the top of the stack.
/PI	Previous Input	Moves the pointer back and displays the previous 20 lines of input.
/PO	Previous Output	Moves the pointer back and displays the previous 20 lines of output.
/PC	Previous Command	Moves the pointer back to the previous FOCUS command issued and displays it and the lines following it. When you enter /PC repeatedly, each command appears above the previously displayed commands. The Session Monitor can display up to 20 lines at one time.
/TOP	Top of the Stack	Moves the pointer to the top of the stack (at the oldest entry) and displays the first 20 lines of the stack.

Command	Command Name	Description
/LN [*n*|1]	Line Command	The effect of /LN *n* varies depending on your previous Session Monitor command: ❏ **After /PP or /TOP,** moves the pointer *n* lines forward and displays the next 20 lines. ❏ **After /PI,** moves the pointer *n* input lines forward and displays the next 20 input lines. ❏ **After /PO,** moves the pointer *n* output lines forward and displays the next 20 output lines. ❏ **After /PC,** moves the pointer *n* lines forward and displays the next 20 lines of commands. **Note:** FOCUS does not recognize the /LN command unless you first issue another Session Monitor display command

Example: Displaying Lines From the Session Monitor Stack

Consider the following terminal session:

```
SET SM = ON
>
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5;
END
>
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END

NUMBER OF RECORDS IN TABLE=       18  LINES=     18

PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY

PAGE     1


COUNTRY     CAR                       DCNEW         RCNEW
-------     ---                       -----         -----
ENGLAND     JAGUAR                 37,135.00     44,390.00
            JAGUAR                 55,970.00     67,455.00
            JENSEN                 74,700.00     89,250.00
            TRIUMPH                21,460.00     25,500.00
FRANCE      PEUGEOT                23,155.00     28,050.00
ITALY       ALFA ROMEO             28,300.00     34,100.00
            ALFA ROMEO             28,300.00     34,100.00
            ALFA ROMEO             24,575.00     29,625.00
            MASERATI              125,000.00    157,500.00
JAPAN       DATSUN                 13,130.00     15,695.00
            TOYOTA                 14,430.00     16,695.00
W GERMANY   AUDI                   25,315.00     29,850.00
            BMW                    29,000.00     29,700.00
            BMW                    30,000.00     31,775.00
            BMW                    50,000.00     68,760.00
            BMW                    55,000.00     70,615.00
            BMW                    41,500.00     45,485.00

PAGE     2


COUNTRY     CAR                       DCNEW         RCNEW
-------     ---                       -----         -----
W GERMANY   BMW                    42,000.00     47,475.00
```

Issue the Session Monitor command /PI to show stacked input lines. The following displays:

```
===>                                      FIRST   INPUT   LINE
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5;
END
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
===>                                      CURRENT INPUT   LINE
```

Issue the Session Monitor command /LN 1 and then /PO to show stacked output lines. The following displays:

```
>
 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY


>
===>                                      CURRENT OUTPUT  LINE
```

Note that the report output does not display because it was displayed using HotScreen.

Issue the request again with SET SCREEN = OFF:

```
SET SCREEN = OFF
>
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAGE     1


 COUNTRY      CAR                      DCNEW         RCNEW
 -------      ---                      -----         -----
 ENGLAND      JAGUAR                37,135.00     44,390.00
              JAGUAR                55,970.00     67,455.00
              JENSEN                74,700.00     89,250.00
              TRIUMPH               21,460.00     25,500.00
FRANCE        PEUGEOT               23,155.00     28,050.00
ITALY         ALFA ROMEO            28,300.00     34,100.00
              ALFA ROMEO            28,300.00     34,100.00
              ALFA ROMEO            24,575.00     29,625.00
              MASERATI             125,000.00    157,500.00
JAPAN         DATSUN                13,130.00     15,695.00
              TOYOTA                14,430.00     16,695.00
W GERMANY     AUDI                  25,315.00     29,850.00
              BMW                   29,000.00     29,700.00
              BMW                   30,000.00     31,775.00
              BMW                   50,000.00     68,760.00
              BMW                   55,000.00     70,615.00
              BMW                   41,500.00     45,485.00
              BMW                   42,000.00     47,475.00
```

Now issue the /PO command. You can see that the report output was appended to the stack:

```
             JAGUAR              55,970.00     67,455.00
             JENSEN              74,700.00     89,250.00
             TRIUMPH             21,460.00     25,500.00
 FRANCE      PEUGEOT             23,155.00     28,050.00
 ITALY       ALFA ROMEO          28,300.00     34,100.00
             ALFA ROMEO          28,300.00     34,100.00
             ALFA ROMEO          24,575.00     29,625.00
             MASERATI           125,000.00    157,500.00
 JAPAN       DATSUN              13,130.00     15,695.00
             TOYOTA              14,430.00     16,695.00
 W GERMANY   AUDI                25,315.00     29,850.00
             BMW                 29,000.00     29,700.00
             BMW                 30,000.00     31,775.00
             BMW                 50,000.00     68,760.00
             BMW                 55,000.00     70,615.00
             BMW                 41,500.00     45,485.00
             BMW                 42,000.00     47,475.00


>
===>                                      CURRENT OUTPUT   LINE
```

Issue the /PC command to display the most recent FOCUS command lines stacked:

```
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
===>                                      CURRENT INPUT    LINE
```

Issue the /PC command again to go back to the previous FOCUS command line stacked:

```
SET SCREEN = OFF
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
===>                                      CURRENT INPUT    LINE
```

Issue the /PC command again. The following lines display:

```
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5;
END
SET SCREEN = OFF
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
===>                                      CURRENT INPUT    LINE
```

Saving Session Monitor Lines

How to:

Save Session Monitor LIne

Example:

Saving Stacked Lines

You can store the contents of the Session Monitor stack in a sequential file (record length 80 bytes).

Syntax: How to Save Session Monitor LIne

To store part of the stack contents in a file, move the pointer to the first line you want to save, and issue the following command

```
/SAVE [n] [filename]
```

where:

n

Is the number of lines in the stack you want to save.

filename

Is the name of the file where the stack is stored. If no name is specified, the default name is FOCSAVE.

Starting at the pointer, the SAVE command saves *n* lines of input, output, or both, depending on the display command issued prior to the SAVE command:

Previous Command	Effect of SAVE Command
/PP or /TOP	Stores both input and output lines.
/PI	Stores only input lines.
/PO	Stores only output lines.
/PC	Stores only command lines.
/LN	The type of lines stored depends on the previous display command, as described in *Session Monitor Display Commands* on page 341.

If the pointer is at the bottom of the stack, FOCUS does not recognize the /SAVE command. FOCUS does recognize /SAVE commands in FOCEXECs.

Once the file is saved, you can edit the file using the system editor (which you can invoke using the IEDIT command) or the TED editor. If you are using CMS and want to print a hardcopy of the file, use the CMS PRINT command.

Example: Saving Stacked Lines

Consider the following terminal session:

```
SET SCREEN = OFF
>
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5
END
>
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
END
END
 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAGE     1


 COUNTRY      CAR                       DCNEW         RCNEW
 -------      ---                       -----         -----
 ENGLAND      JAGUAR                 37,135.00     44,390.00
              JAGUAR                 55,970.00     67,455.00
              JENSEN                 74,700.00     89,250.00
              TRIUMPH                21,460.00     25,500.00
 FRANCE       PEUGEOT                23,155.00     28,050.00
 ITALY        ALFA ROMEO             28,300.00     34,100.00
              ALFA ROMEO             28,300.00     34,100.00
              ALFA ROMEO             24,575.00     29,625.00
              MASERATI              125,000.00    157,500.00
 JAPAN        DATSUN                 13,130.00     15,695.00
              TOYOTA                 14,430.00     16,695.00
 W GERMANY    AUDI                   25,315.00     29,850.00
              BMW                    29,000.00     29,700.00
              BMW                    30,000.00     31,775.00
              BMW                    50,000.00     68,760.00
              BMW                    55,000.00     70,615.00
              BMW                    41,500.00     45,485.00
              BMW                    42,000.00     47,475.00
```

Issue the /TOP command to display lines from the top of the stack:

```
===>                                        FIRST   LINE
>
SET SCREEN = OFF
>
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5
END
>
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

 PAGE     1


 COUNTRY     CAR                        DCNEW          RCNEW
```

Now issue the following commands:

```
/SAVE
TED FOCSAVE
```

The FOCSAVE file contains all of the stacked lines. Issue the command QQ to exit the file without saving it.

Next issue the following commands:

```
/TOP
/PC
/SAVE
TED FOCSAVE
```

The FOCSAVE file contains the FOCUS commands issued:

```
SET SCREEN = OFF
DEFINE FILE CAR
DCNEW/D10.2 = DEALER_COST * 5;
RCNEW/D10.2 = RETAIL_COST * 5
END
TABLE FILE CAR
PRINT CAR DCNEW RCNEW
BY COUNTRY
END
TED FOCSAVE
```

Issue the command QQ to exit the file without saving it.

Next issue the following commands:

```
/TOP
/PO
/SAVE
TED FOCSAVE
```

The FOCSAVE file contains the FOCUS output:

```
+>
+>
+>
0 NUMBER OF RECORDS IN TABLE=       18  LINES=     18

1 PAGE     1


  COUNTRY     CAR                      DCNEW         RCNEW
  -------     ---                      -----         -----
  ENGLAND     JAGUAR                37,135.00     44,390.00
              JAGUAR                55,970.00     67,455.00
              JENSEN                74,700.00     89,250.00
              TRIUMPH               21,460.00     25,500.00
  FRANCE      PEUGEOT               23,155.00     28,050.00
  ITALY       ALFA ROMEO            28,300.00     34,100.00
              ALFA ROMEO            28,300.00     34,100.00
              ALFA ROMEO            24,575.00     29,625.00
              MASERATI             125,000.00    157,500.00
  JAPAN       DATSUN                13,130.00     15,695.00
              TOYOTA                14,430.00     16,695.00
  W GERMANY   AUDI                  25,315.00     29,850.00
              BMW                   29,000.00     29,700.00
              BMW                   30,000.00     31,775.00
              BMW                   50,000.00     68,760.00
              BMW                   55,000.00     70,615.00
              BMW                   41,500.00     45,485.00
              BMW                   42,000.00     47,475.00


+>
+>
+>
```

Issue the QQ command to exit the file without saving it.

Transferring FOCUS Commands to the TED Editor

You can transfer FOCUS commands stored in the Session Monitor stack directly to the TED editor.

To do this, move the pointer (using the commands listed in *Session Monitor Display Commands* on page 341 to the first command you want to edit and enter:

```
/TED
```

This transfers control to the TED editor, which displays the commands from the pointer to the end of the stack. After editing the commands, you can store them as a FOCEXEC called FOCSAVE by entering the TED command FILE.

Note that if the procedure already exists, the FOCUS commands in the stack replace the present contents of the file.

Index

Symbols

A

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Reader Comments

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

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services - Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: http://www.informationbuilders.com/bookstore/derf.html

Name:_____

Company:_____

Address:_____

Telephone:_____Date:_____

E-mail:_____

Comments:

Reader Comments

Information Builders, Inc. Two Penn Plaza, New York, NY 10121-2898 (212) 736-4433

Overview and Operating Environments
Version 7.6

DN1001055.0708