Information Builders

FOCUS for S/390

Overview and Operating Environments
Version 7.2

DN1001055.1101

Preface

This documentation describes how to use FOCUS® Version 7.2 in the VM/CMS and MVS environments. It is intended for all FOCUS users. This manual is part of the FOCUS for S/390 documentation set.

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

The documentation set consists of the following components:

- The *Creating Reports* manual describes FOCUS Reporting environments and features.
- The *Describing Data* manual explains how to create the metadata for the data sources that your FOCUS procedures will access.
- The *Developing Applications* manual describes FOCUS Application Development tools and environments.
- The *Maintaining Databases* manual describes FOCUS data management facilities and environments.
- The *Using Functions* manual describes internal functions and user-written subroutines.
- The *Overview and Operating Environments* manual contains an introduction to FOCUS and FOCUS tools and describes how to use FOCUS in the VM/CMS and MVS (OS/390) environments.

The users' documentation for FOCUS Version 7.2 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		Contents
1	*Introduction to FOCUS*	Provides an overview of FOCUS.
2	*Editing Files With TED*	Presents the FOCUS text editor, TED, and shows how to edit text and data sources.
3	*Terminal Operator Environment*	Describes the Terminal Operator Environment, which provides an optional windowed environment for running FOCUS.

Chapter		Contents
4	*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 different FOCUS facilities interact with the operating system.
5	*OS/390 and MVS Guide to Operations*	Is a guide to operations for FOCUS users who run under MVS. It describes how to allocate files, describes all FOCUS application, extract, and work files, and discusses how different FOCUS facilities interact with the operating system.
6	*Using FOCUS as a Client to an iWay Server*	Describes FOCUS client/server computing and elements of Information Builders' Middleware Technology.

Summary of New Features

The new FOCUS features and enhancements described in this documentation set are listed in the following table.

New Feature	Manual	Chapter
Field-based Reformatting	*Creating Reports*	Chapter 1, *Creating Tabular Reports*
Increased Report Width	*Creating Reports*	Chapter 1, *Creating Tabular Reports*
ACROSS-TOTAL	*Creating Reports*	Chapter 4, *Sorting Tabular Reports*
Tiles	*Creating Reports*	Chapter 4, *Sorting Tabular Reports*
DEFINE FILE SAVE and DEFINE FILE RETURN	*Creating Reports*	Chapter 6, *Creating Temporary Fields*
Forecast	*Creating Reports*	Chapter 6, *Creating Temporary Fields*
Creating Comma-Delimited Files	*Creating Reports*	Chapter 11, *Saving and Reusing Report Output*
Creating Tab-Delimited Files	*Creating Reports*	Chapter 11, *Saving and Reusing Report Output*
Long Master File Names	*Creating Reports*	Chapter 11, *Saving and Reusing Report Output*
JOIN WHERE	*Creating Reports*	Chapter 13, *Joining Data Sources*

New Feature	Manual	Chapter
KEEPDEFINES	*Creating Reports*	Chapter 13, *Joining Data Sources*
Long Master File Names	*Describing Data*	Chapter 1, *Understanding a Data Source Description*
4K Alpha Fields	*Describing Data*	Chapter 4, *Describing an Individual Field*
Extended Currency Symbol Support	*Describing Data*	Chapter 4, *Describing an Individual Field*
SUFFIX = COMT/COMMA/TABT	*Describing Data*	Chapter 5, *Describing a Sequential, VSAM, or ISAM Data Source*
AUTODATE	*Describing Data*	Chapter 6, *Describing a FOCUS Data Source*
CDN	*Developing Applications*	Chapter 1, *Customizing Your Environment*
CENT-ZERO	*Developing Applications*	Chapter 1, *Customizing Your Environment*
Exit on Error	*Developing Applications*	Chapter 1, *Customizing Your Environment*
KEEPDEFINES	*Developing Applications*	Chapter 1, *Customizing Your Environment*
PCOMMA	*Developing Applications*	Chapter 1, *Customizing Your Environment*
Unlimited -INCLUDEs	*Developing Applications*	Chapter 2, *Managing an Application With Dialogue Manager*
SQUEEZ Function	*Using Functions*	Chapter 3, *Character Functions*
STRIP Function	*Using Functions*	Chapter 3, *Character Functions*
TRIM Function	*Using Functions*	Chapter 3, *Character Functions*
DYNAM ALLOC LONGNAME	*Overview and Operating Environments*	Chapter 5, *OS/390 and MVS Guide to Operations*

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
THIS TYPEFACE	Denotes a command that you must enter in uppercase, exactly as shown.
this typeface	Denotes a value that you must supply.
{ }	Indicates two choices. You must type one of these choices, not the braces.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices, not the symbol.
[ ]	Indicates optional parameters. None of them is required, but you may select one of them. Type only the information within the brackets, not the brackets.
underscore	Indicates the default value.
...	Indicates that you can enter a parameter multiple times. Type only the information, not the ellipsis points.
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

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

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

Customer Support

Do you have questions about 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 number (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, http://www.informationbuilders.com. It connects you to the tracking system and known-problem repository 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 number (*xxxx.xx*).
- The FOCEXEC procedure (preferably with line numbers).
- Master File with picture (provided by CHECK FILE).
- Run sheet (beginning at login, including call to FOCUS), containing the following information:
  - ? RELEASE
  - ? FDT
  - ? LET
  - ? LOAD
  - ? COMBINE
  - ? JOIN
  - ? DEFINE
  - ? STAT
  - ? SET/? SET GRAPH
  - ? USE
  - For MVS, ? TSO DDNAME
  - For VM, CMS QFI
- 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 databases, have you tried executing a query containing just the code to access the database?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, http://www.informationbuilders.com.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

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

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

Contents

CHAPTER 1

Introduction to FOCUS

Topics:

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

This chapter introduces FOCUS, outlines its user base, and details its components and facilities.

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 pick up enough background in a three-day class 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 file definition tasks. These facilities present formatted screens from which you select options to create reports and describe files.

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 Interfaces. These Interfaces may access data sources created by other systems or facilities or may allow you to access FOCUS files from programs written in other programming languages.

The FOCUS Language

It is worth discussing briefly at the outset what the terms procedural and non-procedural mean. The basic distinction is this: 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 file, 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 file.
2. Sort the file (by DEPARTMENT).
3. Read a record. If 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. Summary—write report totals.
10. Close files and stop.

The same request in FOCUS might read:

```
TABLE FILE filename
PRINT 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 the Dialogue Manager.

The procedures combining 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. We speak of the TABLE environment, for example, when talking about the commands for requesting tabular reports, or the MODIFY environment for 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. There is a window to:

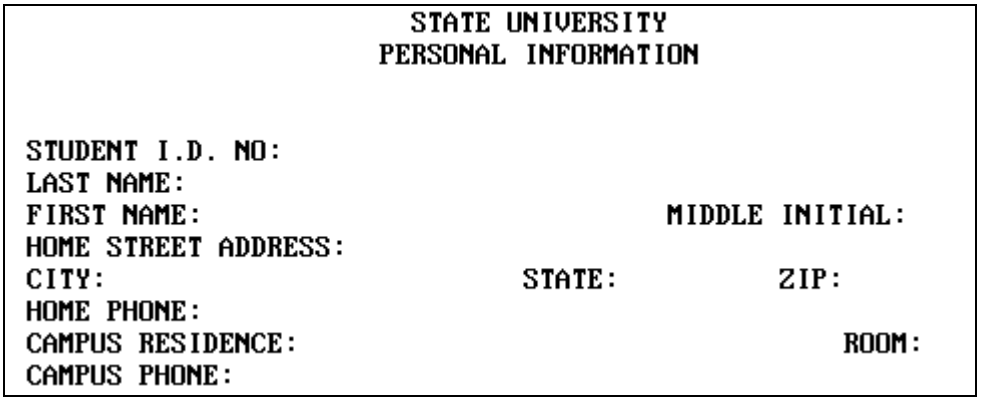
- Accept FOCUS commands you enter at the keyboard.
- Display your FOCUS session log (a list of commands you entered as well as the FOCUS response to each of them).
- Keep a list of every command you enter for later editing or reuse.
- Redisplay your most recently generated report.
- Display a window of current program function (PF) keys 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 3, *Terminal Operator Environment.*

FOCUS Concepts

Your company probably acquired FOCUS because there is 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 the information that your company maintains.

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

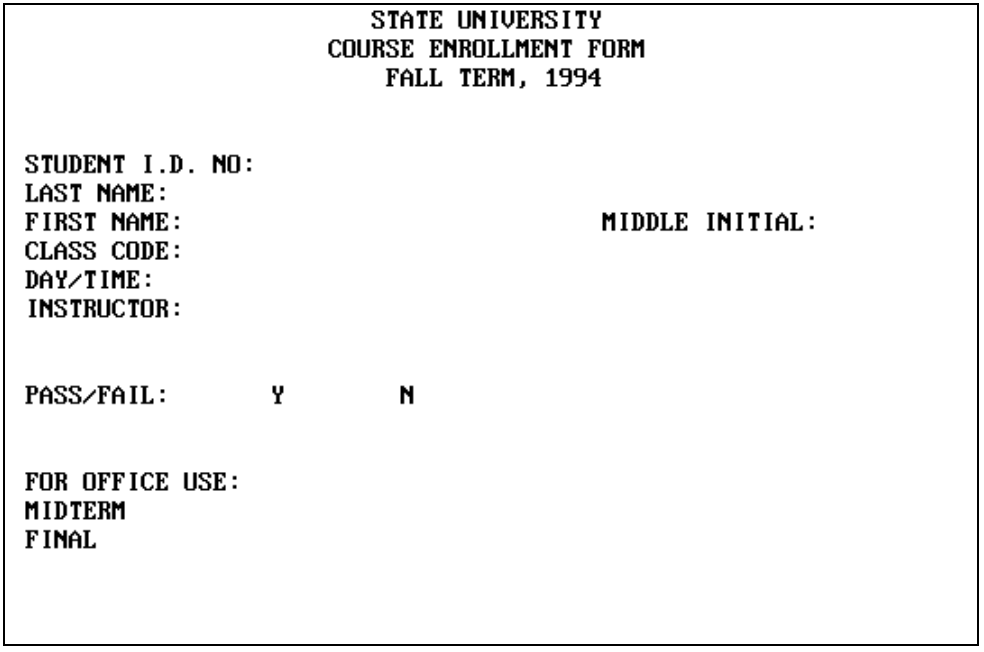


The items of information on the screen are called entry fields. Each field must have a name that identifies it. For example, the last name field might be named LAST_NAME, and street address named STREET. Additionally, each field must be assigned a format to tell the computer whether it is numeric (contains 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 ZIP code).

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 file. Thus, the collected personal data for all of the students at State University could be gathered in a single-segment FOCUS file.

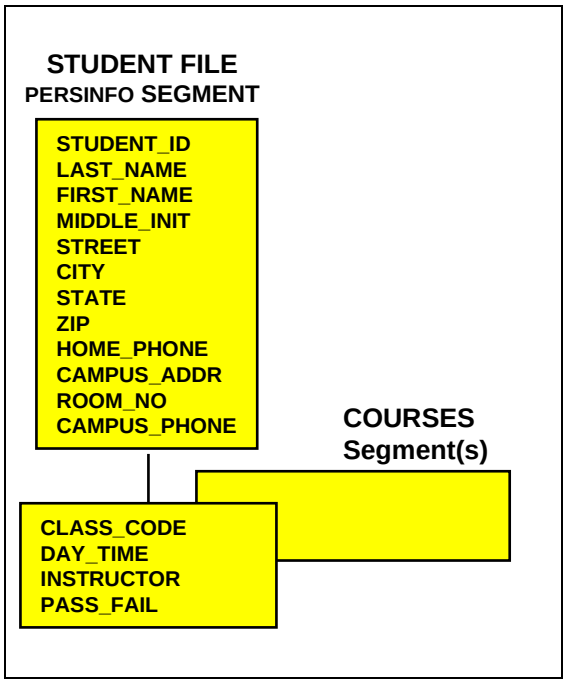
In simple applications, files may consist of a single segment, but generally more than one segment is needed. Consider an additional screen that captures information about each course that a student selects:



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 file or by creating a second file.

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 file for each student.

The above diagram shows the relationship between fields and segments in FOCUS database files. You describe this to FOCUS with a Master File, in which you name the file and each of its segments. Within 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 file.

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

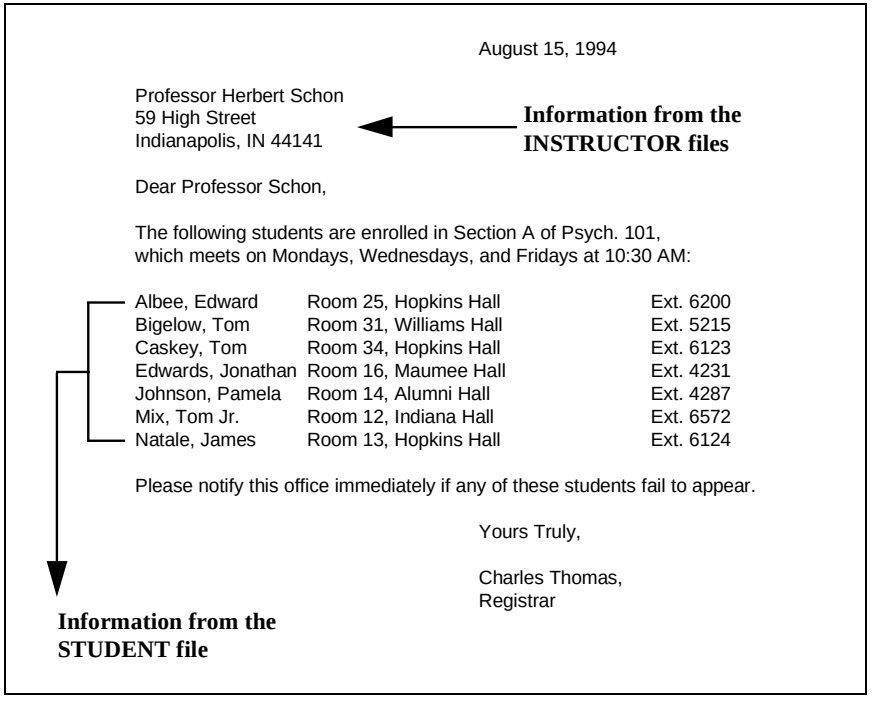
The data (the actual pieces of information described by the entries in the Master File, such as a student's name and Student Identification Number) exist in logical records in the data file. For example, all of the information about a student in the STUDENT file is in a single logical record. Obviously,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

FOCUS also includes facilities for joining files together that enable you to include data from several related databases in a report. For example, suppose State University keeps an INSTRUCTORS file 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 files: the students' names, addresses, and telephone numbers from one file, and the instructor's address from the other. This is accomplished through a JOIN operation in which a common field in both files (in this case, INSTRUCTOR) is used to link the two files.

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



Joined databases remain physically separate, but FOCUS treats them as a single database structure. The JOIN command thus provides a powerful facility for relating files. 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 files 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 files that can be linked.

Features for End Users

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

The following topics are particularly applicable for end users:

- The Report Writer, which works with existing FOCUS and non-FOCUS files. See the *Creating Reports* manual.
- The Financial Modeling Language facility for creating row-oriented financial reports. See the *Creating Reports* manual.
- A full-screen text editor (TED) for creating and saving requests, Master Files, and other text files. See Chapter 2, *Editing Files With TED.*
- Dialogue Manager, a facility for designing and managing applications. See the *Developing Applications* manual.
- The 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 IBM 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.

The Report Writer: TABLE

The FOCUS 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.

Chapter 4, *CMS Guide to Operations*, describes the elements of FOCUS report requests. Such requests always begin with the TABLE command, which invokes the Report Writer.

The file named in your request can be a FOCUS file, a collection of files 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 files. Master Files for non-FOCUS and FOCUS files are described in the *Describing Data* manual.

Within the TABLE environment, 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 file 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 a system editor. These named, executable FOCUS requests are called FOCEXECs (see the *Developing Applications* manual).

Row-oriented Financial Reports: Financial Modeling Language

FOCUS Financial Modeling Language (FML), formerly known as EMR, is an extension of the TABLE environment 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.

The 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 FOCUS GRAPH syntax.

The Text Editor: TED

An optional full-screen editor (TED) is available for creating and editing text files for use inside or outside of the FOCUS environment. Within 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 editor is described in Chapter 2, *Editing Files With TED*.

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 within 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 environment 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-form data files with TED. Note, however, that you cannot use it to edit the data in FOCUS databases. (Use Maintain, MODIFY, or FSCAN to add or edit data in FOCUS databases.)

Data Export Interface

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

- Prepare files 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 file and Master File by extracting request output from FOCUS or external files in FOCUS format.

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

Features for Application Developers

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

- File security features that offer security at every level, from the file itself down to specifying protection for specific values within fields. See the *Describing Data* manual.
- The Dialogue Manager environment for building reusable FOCUS requests (FOCEXECs), including facilities for variable substitution, testing and branching, and reading from or writing to the terminal. See the *Developing Applications* manual.
- Facilities for designing menus and windows to select, enter, and display data. See the *Developing Applications* manual.
- The database management facilities for loading and maintaining files. See the *Maintaining Databases* manual.
- Facilities for designing full-screen data-entry forms, including two screen painters. See the *Maintaining Databases* manual for the FOCUS Screen Painter and the Winform Painter.
- An online, interactive, full-screen database editor (FSCAN). See the *Maintaining Databases* manual.
- Interfaces to other types of databases, including ADABAS, CA-DATACOM/DB®, DB2®, SQL/DS™, Oracle®, Teradata®, CA-IDMS®/DB, IMS/DB, MODEL 204®, SUPRA®, SYSTEM 2000, TOTAL®, and Millennium®.
- The Host Language Interface for reading FOCUS files from programs in other languages.
- User-written subroutines for using subroutines written by other users. See the *Using Functions* manual.

Database Security

Access to data in FOCUS files 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 file:

- 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 within fields.

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

Dialogue Manager

You can enter and execute FOCUS requests at the terminal, or as text files on 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.
- Branching to another area of the procedure or executing nested procedures.
- Reading from, or writing to, 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, via full-screen entry forms, in SET statements, 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.

Database Management: Maintain and MODIFY

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

Maintain

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 window-like forms. You can also include check boxes and buttons to trigger procedures, and display and edit several sets of data at the same time, moving through each set using automatically-provided scroll bars. The Painter generates these window-like 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

The MODIFY command in FOCUS invokes the data management environment, which provides complete facilities for the following data file maintenance activities:

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

The *Maintaining Databases* manual describes how to incorporate these facilities into MODIFY requests that you can execute to update a FOCUS database. Such requests can range from simple ones containing a few instructions, to extensive procedures 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 file maintenance procedures:

- The MODIFY environment supplies the file and record handling facilities, and validation and calculation features.
- The Dialogue Manager environment provides control facilities for creating MODIFY procedures that can include variable fields and prompt for data.
- The MODIFY subcommand CRTFORM passes you to the FIDEL environment 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 files, or you can use the COMBINE command in MODIFY requests to update multiple files simultaneously.

FOCUS offers two operating environments for file maintenance:

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

Full-Screen Data Entry Forms: FIDEL

File maintenance and Dialogue Manager procedures require full-screen data entry forms for entering information needed to update files. The facility for describing screen forms is the FIDEL environment that is discussed in the *Maintaining Databases* manual. (FIDEL stands for FOCUS Interactive Data Entry Language.) 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.)

Database Editor: FSCAN

The FSCAN facility, described in the *Maintaining Databases* manual, is a file maintenance utility that enables you to edit FOCUS databases directly on the screen. FSCAN displays databases as if they were flat files on a full-screen system editor. FSCAN commands allow you to scroll through records, navigate the database, 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 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 files. For more extensive maintenance, use the MODIFY or Maintain facilities.

The 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 databases are used, who uses them, and how resource usage is distributed.

SmartMode is available for MVS 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.

There are also file utilities for performing a variety of file handling tasks: initializing files, concatenating them, rebuilding files and their indexes, joining files together, and transferring files 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 for S/390 Users Manual and Installation Guide.*

CHAPTER 2

Editing Files With TED

Topics:

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

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.

Introduction

This chapter describes 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 databases, 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:

- Commands to move or copy lines of data from one window to another using the split-screen facility.
- Access to 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.

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]
```

Refer to Chapter 4, *CMS Guide to Operations,* for more information.

In MVS, enter:

```
TED FOCEXEC(member)
```

or

```
TED name
```

Refer to Chapter 5, *OS/390 and MVS 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

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 MVS.
- 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 MVS is:

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

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



====>
                                                            TED
```

Note: Screens used in this chapter are from a CMS operating system; the only difference between a CMS and MVS screen display in TED is the file name.

The 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 2-11.

The Command Line

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

Moving the Cursor

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

↓	Moves the cursor down.
↑	Moves the cursor up.
→	Moves the cursor to the right.
←	Moves the cursor to the left.
Tab	Moves the cursor to the next line in TYPE or to the next Tab stop in EDIT.
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 2-42). TED enables you to create lines up to 160 characters long in CMS (159 in MVS). 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 2-8.

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 left-most 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 is displayed:

```
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 2-8.

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

```
PAINT
```

at the command line (or press PF4). TED will scan 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, place the number of the CRTFORM next to PAINT. For example

```
PAINT 3
```

will access the third CRTFORM from the current line.

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 2-8).

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

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

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

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

Note:

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

Adding Lines

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

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

ADD

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

`Add` *n*

where:

n

Is any number of lines you are adding.

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

```
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

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 you can 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 10, 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 TENTH PHYSICAL LINE OF THE SCREEN.
===== * * * END OF FILE * * *


====>
                                                      EDITING MODE
```

Inserting and Replacing Text

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

Command Line Commands

```
REplace
Overlay
Input
```

REplace

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

```
REplace string
```

where:

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

Overlay

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

```
Overlay string
```

where:

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

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

For example:

```
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 ABOVEBUT NOT THIS PART
THIS IS AN EXAMPLE OF THE OVERLAY COMMAND
* * * END OF FILE * * *




====>
                                                    EDITING MODE
```

Input

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

```
Input string
```

where:

string
Is the text you want placed after the current line.

Deleting and Recovering Deleted Text

The following commands delete or recover deleted text:

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

==D==

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

You can also use the syntax

==D*n*=

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 (no alphabetics or numerics) 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

If 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 will appear 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

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, FOrward, PF7, PF8, PF19, PF20

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.

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.

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 you want to scroll to.

±n

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

Locating and Changing Text

When viewing a file that you wish to modify, you can either move the cursor to the lines to be edited and type over the text, or you can 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 for LOCATE 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 2-35). 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 will change 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.
===== * * * END OF FILE * * *




====>
                                                        EDITING MODE
```

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

Copying and Moving Text

The following commands duplicate and move text in a file:

Command Line Commands	Prefix Area Commands
COpy	==C== ==CC= ==" "=
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 will be duplicated immediately after its present position.

==“ n=

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

DUplicat

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

```
DUplicat n m
```

where:

n

Is the number of duplications.

m

Indicates how many lines are included in the duplication.

MOve

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

```
MOve n m
```

where:

n

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

m

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

==M==

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

=MM==

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

Joining and Splitting Text

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

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

==J==

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

```
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

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

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 MVS, 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 is displayed:

```
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 MVS, 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 2-28.
3. Retrieve the lines from temporary storage using the GET command.

Displaying a Scale and Line Numbers

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

Command Line Commands

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

NUm ON

To replace the prefix area with numbers, enter

```
NUm ON
```

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

```
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 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

The following commands control uppercase or lowercase text in a file

Command Line Commands

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

CAse M

To have uppercase and lowercase characters enter

```
CAse M
```

at the command line.

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

CAse U

To get only uppercase characters enter:

```
CAse U
```

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

UPPercas

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

```
UPPercas n
```

where:

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

LOwercas

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

```
LOwercas n
```

where:

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

Ending a TED Session

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

Command Line Commands	Function Keys
Quit	PF3
QQuit	
FILe	
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 MVS.

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 will cause an error.

Syntax Summary

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

```
TYPE
EDIT
INPUT
PAINT
```

Function Keys

The following list shows the PF Key assignments in TED.

Key	Action
PF1, PF13	Displays meaning of the keys and help information.
PF2	Inserts line after 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 LEFT one page.
PF11, PF23	Moves to 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 delete 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.

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

Command Repeat. 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 (MVS).
`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 (MVS).

Command	Action
`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 (MVS).
`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 MVS/TSO command.
`Next n`	Moves forward *n* lines.
`NUmber ON/OFF`	Sets up prefix area with numbers.
`Overlay string`	Overlays string on current line; existing text remains after end of string.
`PAINT n`	Paints the “nth” CRTFORM.
`PFnn string`	Sets PF key *nn* to the specified string.
`Put n [filename]`	Puts *n* lines to specified file.
`PUTD n [filename]`	See PUT, but lines are deleted.

Command	Action
`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 right or *n* columns.
`RIGHTP`	Moves one-half screen to 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 (MVS users only).
`TEd` *`filename`*	Edits another file from within the current file (CMS).
`TED` *`ddname(member)`*	Edits another file (MVS).
`Top`	Goes to top of file.
`TSO` *`command`*	Issues TSO/MVS command (MVS 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.

Command	Action
`=`	Repeats last command.
`?`	Shows last command.
`? n`	Shows text of error message number *n*.
`?F filename`	Shows fields in file *filename*.

CHAPTER 3

Terminal Operator Environment

Topics:

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

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

Introduction

To access the Terminal Operator Environment, issue the WINDOW ON command from the FOCUS command level. 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.

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:

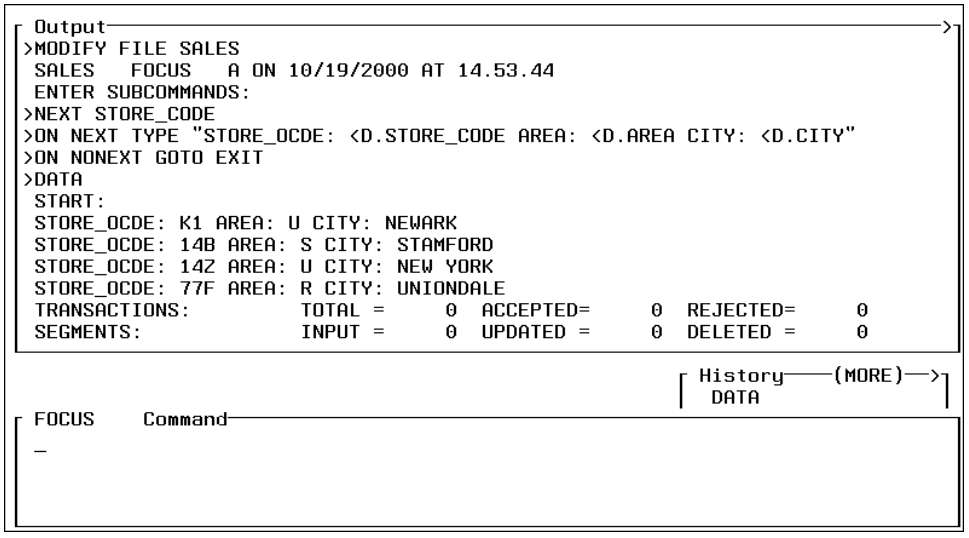
Window	Function
Command	Accepts user input: FOCUS commands and requests, operating commands, and WINDOW commands.
Output	Displays input and resulting output.
History	Records commands and requests, so you can recall them and resubmit them later in the session.
Help	Displays function key settings. You may redefine settings by typing over the existing ones.
Error	Displays FOCUS error messages.
Table	Displays the most recent TABLE request.
Fields	Displays a list of available fields, so you can select one and continue entering a request.

This chapter discusses:

- How to invoke the Terminal Operator Environment.
- How to activate a window.
- The seven types of windows (Command, Output, History, Help, Error, Table, and Fields) and their functions.
- Program function keys and how to reassign them.
- WINDOW commands.

This chapter also uses the SALES and EMPLOYEE databases in its examples. For more information about either database, see the *Creating Reports* manual.

The following is a sample screen illustrating 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 pre-defined 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.

To make the Terminal Operator Environment your FOCUS default environment, include the WINDOW ON command in your PROFILE FOCEXEC.

Invoking the Terminal Operator Environment

To enter the Terminal Operator Environment, type the following command from the FOCUS command level or include it 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. The syntax is

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

where:

`ON`

Invokes the Terminal Operator Environment.

`OFF`

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

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 display on the screen, only one may be used at a time. This active window appears highlighted. There are four 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 3-16 for WINDOW commands).

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

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

Types of Windows

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

Command Window	Accepts all FOCUS commands, FOCUS requests, and WINDOW commands (*The Command Window* on page 3-5).
Output Window	Displays Command Window input and resulting output; accesses Hot Screen facility (*The Output Window* on page 3-9).
History Window	Lists commands and requests entered in the Command Window (*The History Window* on page 3-9).
Help Window	Displays function key settings; user may redefine settings for the current session (*The Help Window: Revising PF Key Settings* on page 3-10).
Table Window	Provides the most recent TABLE request (*The Table Window* on page 3-11).
Error Window	Displays FOCUS error messages (*The Error Window* on page 3-12).
Fields Window	Displays available fields for a specified file (*The Fields Window* on page 3-12).

The Command Window

All commands and requests are entered at 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 or requests.

First, 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 report, 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.

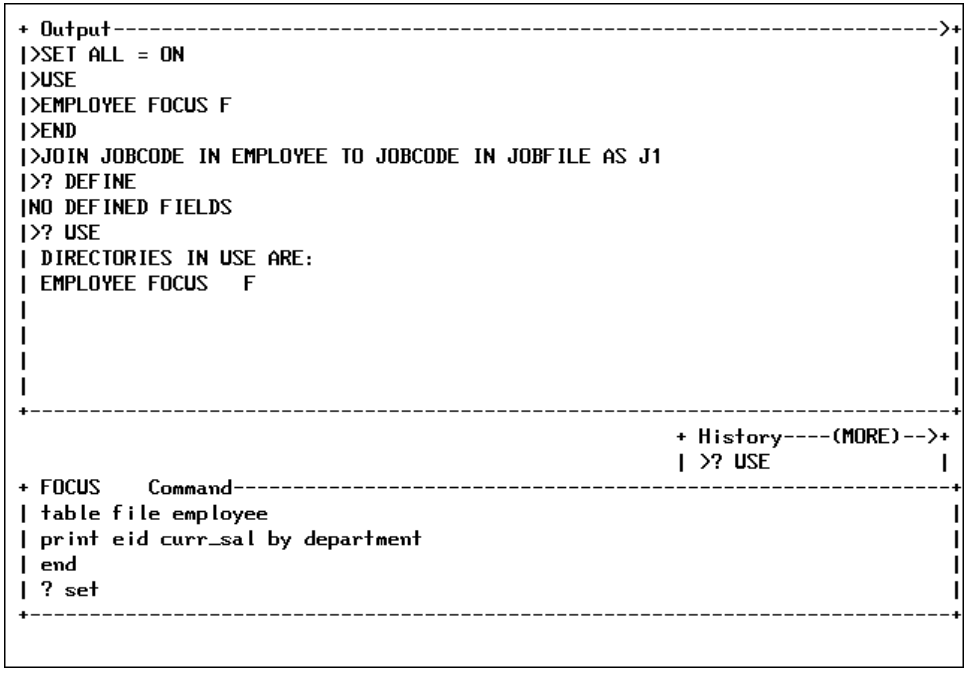
If your request is longer than four lines, press PF2 to enlarge the window. Finish entering the rest of the request; do not press PF2 again. Then, press Enter. The complete request is copied to the Output Window and to the History Window and is submitted for execution.

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.

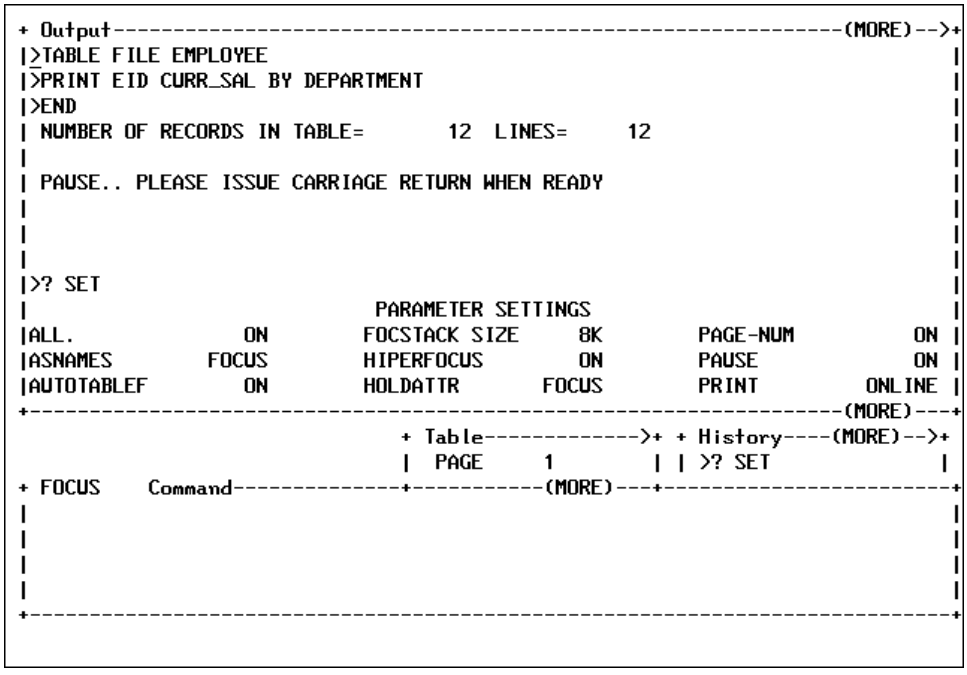
Note: If you type over an existing command or request, be sure to delete leftover characters.

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.

For example, 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.



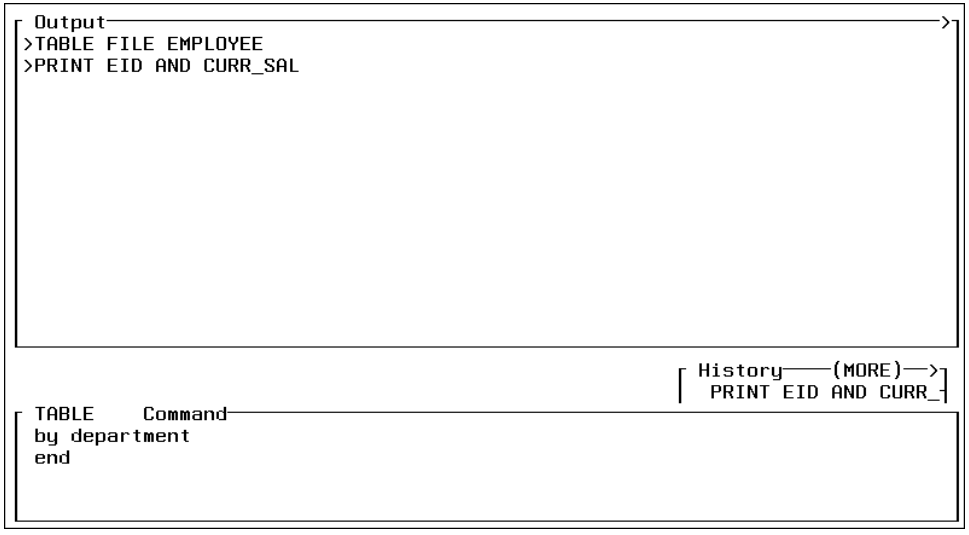
The Command Window also displays the current command mode in a title area. The title area is on the left side of the window's top border. It functions like the prompt does in the default FOCUS environment. In the previous screen sample, for example, the title area displays

```
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.

The 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 *The Table Window* on page 3-11).

Within a TABLE or GRAPH request, you can use the Output Window to display field names, aliases, and formats for the specified file (see *The Fields Window* on page 3-12).

The 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 3-24.

The Help Window: Revising PF Key Settings

The Help Window displays the program function, PA2, and Clear key 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.

For example, 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
```

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 re-enter the optional environment.
- The FOCUS HELP facility is available from this environment; issue the FOCUS HELP command from the Command Window.

The 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.
- Output from reports created with the TABLEF command will not appear in the TABLE window.

The 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.

The Fields Window

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

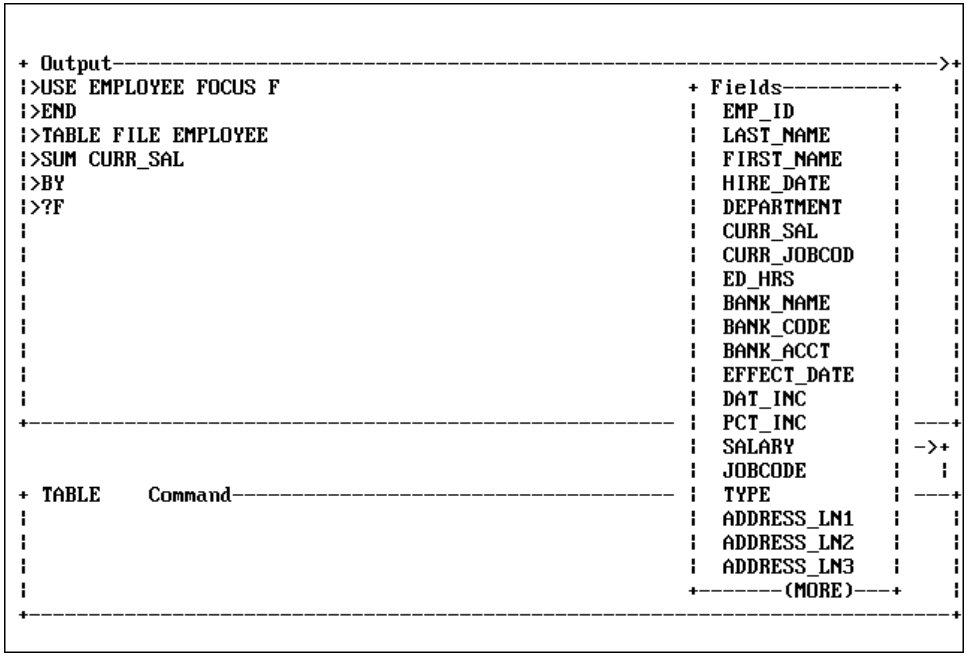
As you enter your request from the Command Window, issue the ?F query command:

```
?F
```

The Fields Window displays, 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 display 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.

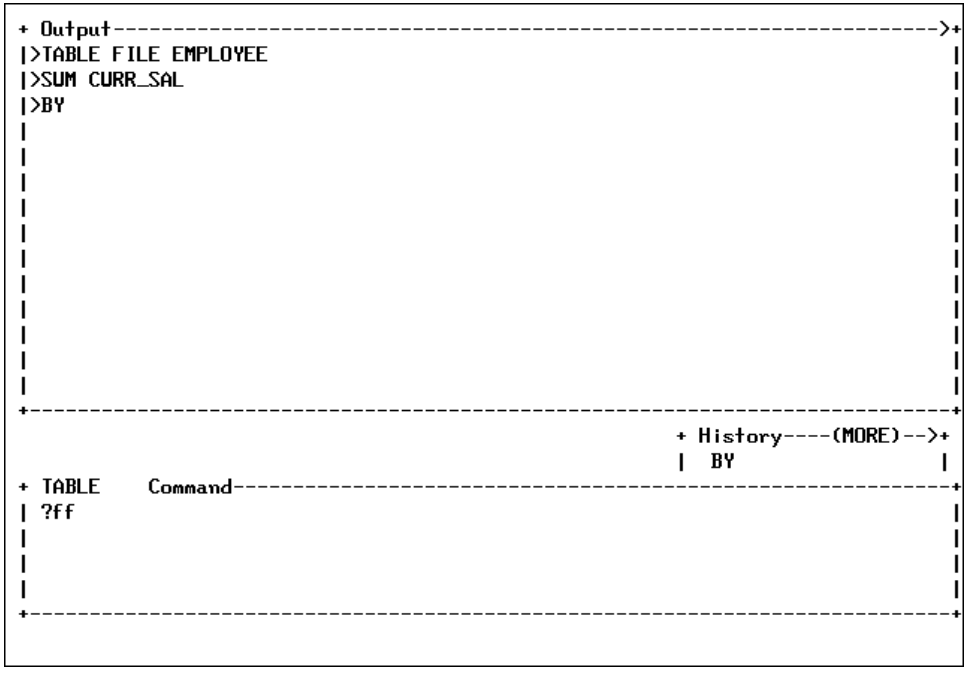
As you enter your request from the Command Window, issue the ?FF query command:

```
?FF
```

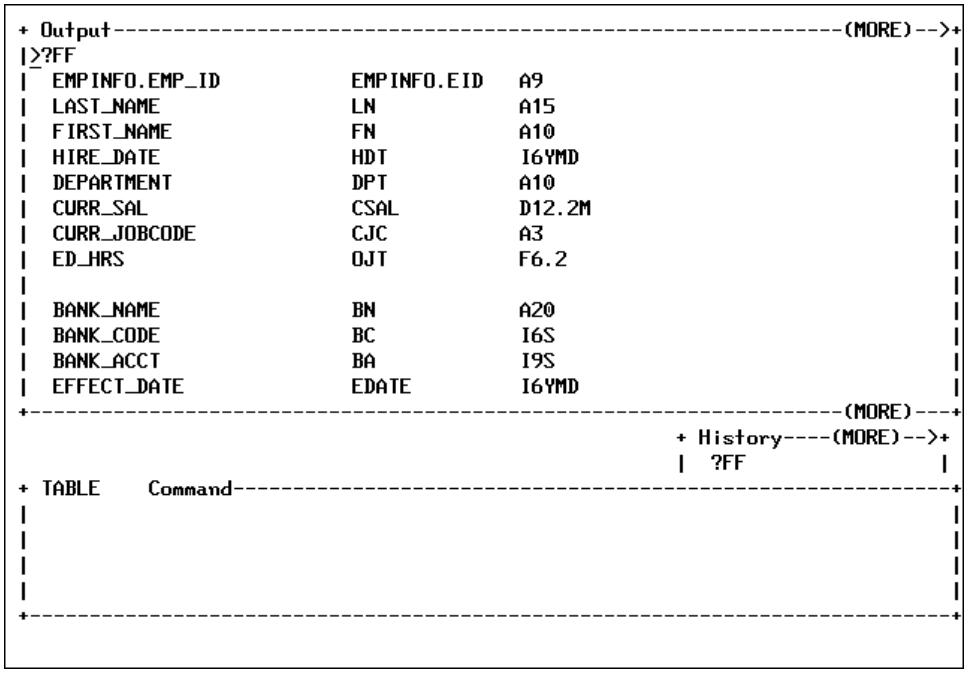
The Output Window displays the fields. Note the field you wish to use in the request and then press the PF12 key to return to the Command Window. Type the field you selected and press Enter.

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 is displayed. 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 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 *The Help Window: Revising PF Key Settings* on page 3-10.) Commands that you use often may be stored in your PROFILE FOCEXEC.

This section contains:

- The ACTIVE and NEXT commands which activate a window.
- The CLEAR command which clears the contents of a window.
- The SET CONTINUE, SET AUTOSCROLL, and SET IMMEDTYPE commands which control the behavior of the Output Window.
- Commands which customize your screen: CLOSE, OPEN, MOVE, SET ERRORS, SET, and SIZE.
- The HELP command which displays the Help Window.
- The ZOOM command which enlarges a window.
- The RECALL command which recalls issued commands.
- The ROUTE command which routes the contents of a window to a file for VM/CMS or to a data set for MVS/TSO.
- The SCROLL command which controls the display of the window contents.

Note: In the examples that follow, 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

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

ACTIVE Command

The ACTIVE command syntax is:

```
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, if you wanted to activate the History Window, you would enter:

```
WINDOW ACTIVE HISTORY
```

NEXT Command

The NEXT command activates the next window in the screen sequence. The syntax is:

```
WINDOW NEXT
```

Pressing PF12 is equivalent to issuing the NEXT command.

Clearing a Window

The CLEAR command erases the contents of a window. The syntax is:

```
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

Two commands control the contents of the Output Window: SET AUTOSCROLL and SET CONTINUE.

SET AUTOSCROLL Command

The SET AUTOSCROLL command syntax is

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

where:

ON
: Is the default; 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.

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, you would enter:

```
WINDOW SET AUTOSCROLL OFF
```

SET CONTINUE Command

The SET CONTINUE command syntax is

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

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.

SET IMMEDTYPE Command

The SET IMMEDTYPE command syntax is

```
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
: Is the default. Buffers all line mode output. The output appears in the Output Window as a new full screen.

Customizing Your Screen

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

```
SET SBORDER=OFF
```

before entering the Terminal Operator Environment.

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

CLOSE Command

The CLOSE command removes a window from the screen. The syntax is:

```
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.

OPEN Command

Conversely, the OPEN command displays a closed window in its normal screen location. 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. The syntax is:

```
WINDOW OPEN [windowname]
```

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

```
WINDOW OPEN HISTORY
```

MOVE Command

The MOVE Command moves a window to a new screen location. The syntax is

```
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.

SET ERRORS Command

The SET ERRORS command controls the length of the error message that displays in the Error Window. The syntax is

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

where:

`SHORT`
: Displays the short form: the error number and description.

`LONG`
: Is the default. Displays the long form: the error number, description, and an explanation.

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

```
WINDOW SET ERRORS SHORT
```

SET Command

The SET command is used to redefine key settings. The syntax is

```
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, specify:

```
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 *The Help Window: Revising PF Key Settings* on page 3-10).
- Key settings change back to their default settings when you end the FOCUS session unless they are defined in the PROFILE FOCEXEC.

SIZE Command

The SIZE command changes the height or width of a window. If you change the height, the bottom of the window is increased or decreased accordingly. If you alter the width, the right window side is increased or decreased accordingly. The syntax is

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

where:

`WIDTH`
Changes the width of the window.

`HEIGHT`
Changes the height of the window.

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

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

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

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

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

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

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

```
WINDOW SIZE HISTORY HEIGHT 5
```

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

Displaying the Help Window

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

```
WINDOW HELP
```

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

Enlarging a Window

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

```
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. Continue to press the PF6 key and previously entered commands appear according to the order in which they were entered.
- Position the cursor in an active History Window next to the command and press Enter.

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

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

Routing Window Contents

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

```
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

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. The syntax is

```
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. The reason is that the automatic autoscroll feature might override your explicit SCROLL command.
- You may also scroll the Output Window forward with the PA2 or CLEAR key while you are working in another active window.

CHAPTER 4

CMS Guide to Operations

Topics:

- Introduction
- Application Files
- Extract and Work Files
- FOCUS Facilities Under CMS

FOCUS for CMS is a fourth generation language and database management system providing decision support capabilities to application developers, analysts, data processing professionals and users with varying levels of experience.

This guide provides information on the features of FOCUS as implemented on the CMS operating system.

Introduction

All of the FOCUS features described in this documentation set are available to you. As a CMS user, you are familiar with the requirements of your particular operating system. This section contains all the 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.

Release statistics, installation and operational changes, and maintenance log information like 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 displays with a choice of reports regarding release specific information. READMEF includes operational notes, install notes, known problems, problems corrected in this release, new features, advisories, Fusion release notes, and license management information.

Referencing Files

In FOCUS, you always reference files by their fileid as follows

```
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 that the file resides on.

The following is a list of the major files that you will use in FOCUS and their functions:

System Files

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.
OFFLINE	Specifies alternate destination for printed reports.

Application Files

MASTER	Master Files.
ACCESS	Access Files. (Optional except for intelligent partitioning. For information, see the *Describing Data* manual.)
FOCEXEC	Stored procedures.
FOCUS	FOCUS databases and external indices.
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/or DBA information for extract files.
WINFORMS	Winforms used in a Maintain procedure.

Extract Files

*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 allocated by you.

Work Files

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 command. The PICKUP command reads it back in.
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

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

```
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

You do not have to explicitly define most of your files prior to referring to them in a FOCUS session. FOCUS will dynamically allocate almost all files.

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

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

Entering 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.

Exiting FOCUS

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

```
FIN
```

Application Files

This section describes how FOCUS references and searches for your application files such as Master Files, FOCEXEC files (stored procedures), FOCUS databases, 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 file, or use a database 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 files to FOCUS. The description of a FOCUS file and all files it cross-references must be available whenever you refer to the data file. Generally, the description and the data reside on the same disk, the A disk, but the 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 files.

The Master File and the data source that it is describing usually have the same file name. All Master Files must consist of 80-byte fixed length records.

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.

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 Files

Stored procedures can be saved under any CMS file name and file type. They are most conveniently filed under the file type FOCEXEC. All FOCEXECs must consist of fixed length, 80-byte records.

You execute a stored procedure by typing

```
EXEC procedurename
```

or

```
EX procedurename
```

where:

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

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:

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

For example, a procedure could be stored as:

```
SALES REPORT A
```

To execute, type:

```
EX 'SALES REPORT'
```

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

Note: The file mode defaults to *. 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.

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

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

The 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 *Entering FOCUS* on page 4-5 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.

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.

FOCUS Databases

FOCUS databases contain data written in FOCUS format. Each database has a file name that matches the file name of the file's Master File and has a file type of FOCUS. For example, if the database's Master File is LEDGER MASTER, then the FOCUS data file is LEDGER FOCUS. You can override these defaults with the USE command, with a DATASET attribute in the Master File, or with an Access File. These techniques are explained in the *Describing Data* manual.

All FOCUS databases have a record length of 4096 and a fixed length record format. The maximum size of a FOCUS database is 2G. Through the use of partitioning, the total size can be up to 500G. See the *Describing Data* manual for information.

External Indices for FOCUS Databases

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

Database Security: ENCRYPT, DECRYPT, and RESTRICT

The ENCRYPT Command

The syntax for the ENCRYPT command in CMS is

```
ENCRYPT FILE filename [filetype [filemode]]
```

where:

filename, *filetype*, and *filemode* make up the fileid of the file you are encrypting. The file type defaults to MASTER; the file mode defaults to A. For example, the command

```
ENCRYPT FILE EMPLOYEE
```

encrypts the file EMPLOYEE MASTER A.

The DECRYPT Command

The syntax for the DECRYPT command in CMS is

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

where

filename, *filetype*, and *filemode* make up the fileid of the file you are decrypting. The file type defaults to MASTER; the file mode defaults to A.

The RESTRICT Command

The syntax of the RESTRICT command for restricting existing files is

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

where:

filename, *filetype*, and *filemode* make up the fileids of the files you are restricting. The file type defaults to FOCUS; the file mode defaults to A. For example, the command

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

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

FUSELIB

The FOCUS User-Written Subroutine Library, FUSELIB, contains additional calculation and utility routines. To access any of these routines issue the CMS command:

```
GLOBAL TXTLIB FUSELIB
```

Do this before you attempt to use one of these routines.

FOCCOMP Files

The fileid of a compiled procedure is

```
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.

The FOCCOMP file is used to run the compiled MODIFY procedure:

```
RUN focexecname
```

Window Files

Window files contain the windows, menus, and related information created by Window Painter. There are three types of window files:

- Compiled window files, which 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 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, which 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. Before they can be executed efficiently by a -WINDOW statement or edited by Window Painter, they must be compiled using the WINDOW COMPILE command.

- Window documentation files, which 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.

These three types of window files are described in the following sections.

Compiled Window Files

The fileid of a compiled window file is

```
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.

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

The fileid of a window transfer file is

```
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.

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.

Note that 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

The fileid of a window documentation file is

```
filename TRF
```

where:

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

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.

Note that 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.

Non-FOCUS Data Sources

You can use the FOCUS query language to read data sources that are not in FOCUS format. FOCUS can read: QSAM, ISAM, VSAM files; IMS, CA-IDMS/DB, ADABAS, TOTAL, SYSTEM 2000, 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 allocated. Note the following example:

```
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 dsname must be the VSAM cluster name. Use the IDCAMS utility to obtain it; the CMS LISTFILE command cannot be used for this purpose.

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
```

Dynamically Setting the VSAM Addressing Mode

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

The syntax is:

```
{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 will not need to change the default.

TRACE Files

To record output from either the TRACE or ECHO option of MODIFY in a file, first allocate the file by entering this CMS command

```
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.

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

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.

The fileids are

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

where:

filename
Is the CMS file name you specify in TableTalk.

To edit saved TableTalk sessions, enter the command:

```
TABLETALK EDIT [filename]
```

If you omit the file name, a list of all TTEDIT files will be displayed. Include a file name to edit a particular saved TableTalk session.

HOLDSTAT Files

HOLDSTAT files enable you to include FOCUS DBA information and environmental comments in HOLD and PCHOLD Master Files. 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.

The fileids for a HOLDSTAT file are

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

where:

fn
Is the file name of your customized HOLDSTAT file.

fm
Is the file mode of the disk.

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

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.

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. WINFORMS records are 80 bytes and fixed length. 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 via 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 and 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. For example, to place all extract files on the D disk, enter:

```
SET TEMP DISK = D
```

Note that changing the TEMP DISK will only affect the disk on which new files are created.

Extract Files

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

Using FILEDEF to Create Extract Files

By default, extract files are written to the 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.

Syntax How to Use 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
```

Note:

- 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.

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

Execute the HOLD command to save the output of a report request

```
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.

Note: 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 file. These files are then used as input to the MODIFY that creates the final FOCUS file. The new FOCUS and Master files 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.

SAVB Files

When you issue the command SAVB, FOCUS saves a report in binary format. That is, all numeric fields are stored in binary format. The file cannot be printed. Also, all character fields are padded with spaces to a multiple of 4 bytes. The syntax is

```
SAVB [AS ddname]
```

where:

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

SAVE Files

When you issue the SAVE command, FOCUS saves all columns in the report in printable, character format with no spaces between columns. The syntax is

```
SAVE [AS ddname]
```

where:

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

Extract Files That You Allocate

LOG Files

To log transactions on a file, first allocate 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.

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
```

Transaction Files

To have a MODIFY request read a transaction file, allocate 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 allocate the file if the file has a file type of DATA, a file mode of A and the records are 80-byte fixed format.

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
```

Sending TYPE Messages to a Transaction File

To send MODIFY TYPE messages to a file, first allocate 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.

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
```

Dialogue Manager Input and Output Files

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

Work Files

FOCSTACK

The 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 allocated as FOCSORT FOCTEMP on the temporary disk. If the FOCSORT work file is allocated to a disk other than the temporary disk, FOCUS respects the allocation.

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 allocated as FOCPOST FOCTEMP.

REBUILD

The REBUILD command options REBUILD and REORG require a work file allocated 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 file 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 databases.

The sequential output file will be allocated 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 *FOCUS for IBM Mainframe 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 allocate 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, 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

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

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

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

FOCUS Facilities Under CMS

This section describes the use of FOCUS facilities in the CMS environment.

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
```

The 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

To enter TED, use the following syntax

```
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).

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.

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.

Changing Filename, Filetype, and Filemode

The following TED commands enable you to change the file name, file type, and the file mode of existing files.

To change the file name, issue

```
FN newfilename
```

or:

```
FILEName newfilename
```

To change the file type, issue

```
FT newfiletype
```

or:

```
FILEType newfiletype
```

To change the file mode, issue

```
FM newfilemode
```

or:

```
FILEMode newfilemode
```

Issuing CMS Commands From TED

To issue a CMS command while in the TED environment, use the syntax:

```
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.

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 ENTER. 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 5 to 15 minutes to print a graph depending on the 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 it 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 Users Manual* for further information.

Allocating Files to Hold Formatted Graphic Output

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

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. To access the FOCUS Menu, issue:

```
EX FMMAIN
```

The FOCUS Menu may also be invoked automatically when you enter FOCUS. To automatically invoke it, edit the file SHELPROF FOCEXEC that is 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 look like:

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

Accessing the FOCUS 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.

To access the FOCUS ToolKit, issue:

```
EX ISHFSHLL
```

The FOCUS ToolKit may also be invoked automatically when the user enters FOCUS. To automatically invoke it, edit the file SHELPROF FOCEXEC, which is found on either the FOCUS production disk or on any accessed disk, to read:

```
EX ISHFSHLL
-EXIT
```

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 'file name ISHFALOC', where file name is the name of the file (as used by FOCUS). For example, the file 'CAR ISHFALOC' accesses the CAR FOCUS file:

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

Accessing 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 nonlinear 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. Power Reporter has a PC Windows look and feel and contains context sensitive help.

To access Power Reporter, issue:

```
EX PWREP
```

National Language Support

FOCUS is designed with consideration for National Language Support (NLS). FOCUS stores data as entered and retrieves the data based on the current code page mapping. FOCUS can be configured for local language messages and keyboards. For installation instructions, refer to the *FOCUS for IBM Mainframe CMS Installation Guide*. Your Information Builders' representative can provide a list of available language choices.

Checking Current Language Settings

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

The syntax is:

```
? LANG
```

FOCUS supports languages such as Kanji that require two bytes of storage per character. See the *Describing Data* manual.

Issuing CMS Commands From Within FOCUS

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

```
CMS LIST * FOCEXEC A
```

issues the CMS LIST command.

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
```

Extended Plists

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 tokenized 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 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.

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

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.

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.

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.

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.

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.

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

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.

Adding and Deleting GLOBAL Libraries

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.

The syntax for adding or deleting libraries is as follows:

```
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.

CHAPTER 5

OS/390 and MVS Guide to Operations

Topics:

- Introduction
- Application Files
- Extract Files
- Work Files
- Calling FOCUS Under TSO
- FOCUS Facilities Under TSO
- The DYNAM Command

As an MVS/TSO user, you should be aware of requirements unique to your operating system. This chapter contains all the 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 MVS/TSO environment.

Introduction

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

Release statistics, installation and operational changes, and maintenance log information like 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 displays with a choice of reports regarding release specific information. READMEF includes operational notes, install notes, known problems corrected in this release, new features, advisories, Fusion release notes, and license management information.

Referencing Files

In FOCUS, you always reference files by ddname rather than by their fully qualified data set names. Data set names are arbitrary but must conform to your installation's naming standards.

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 four categories:

Required DDNAMEs

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

MASTER	Master Files.
ACCESS	Access Files (Optional except for intelligent partitioning. For information, see the *Describing Data* manual.)
FOCEXEC	Stored procedures.
ddname	FOCUS databases and external indices.
USERLIB	Library of user programs.

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/or DBA information for extract files.
WINFORMS	Winforms used in a MAINTAIN procedure.

Extract DDNAMEs

*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

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.
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.

Allocating Files

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 5-5 and *The DYNAM Command* on page 5-61).

The JCL needed to operate FOCUS is usually the same and varies only because of local installation conventions. Generally, the 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.

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: The purpose of each of these files and its parameters are discussed in detail in the next section. These allocation procedures are discussed in more detail in *Calling FOCUS Under TSO* on page 5-33.

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 databases 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 5-6 through *Work Files* on page 5-30 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 re-allocate 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 re-allocate 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 re-allocation 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

This section describes how FOCUS references and searches for your application files such as MASTER files, FOCEXEC files (stored procedures), FOCUS databases, 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 DCB attributes of the MASTER and HOLDMAST PDSs are: RECFM=FB, LRECL=80, with BLKSIZE a multiple of LRECL.

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. 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 DCB attributes of the ACCESS PDS are: RECFM=FB, LRECL=80, with BLKSIZE a multiple of LRECL.

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 name specified as the ACCESS FILE attribute in a Master File. For example, the FOCUS command TABLE FILE CAR requires a member CAR in the ACCESS file PDS.

FOCEXEC Files

All FOCUS procedures can be stored in one or more partitioned data sets allocated to ddname FOCEXEC. The names of the procedures correspond to the member names.

Give the FOCEXEC the following DCB attributes: RECFM=FB, LRECL=80, with BLKSIZE a multiple of LRECL. The line numbering conventions for Master Files apply to FOCEXEC files as well (see *Master Files* on page 5-7).

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. In batch mode, you can create FOCEXECs with the utility programs IEBPTPCH and IEBUPDTE.

Executing FOCEXECs

You can execute a FOCEXEC procedure by issuing

```
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.

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
```

Several partitioned data sets of FOCEXEC procedures may be concatenated under the ddname FOCEXEC.

The 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 5-33.

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 and the logical record length (LRECL) is 80 bytes, 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.

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
/*
```

Note:

- Do not use the -RUN statement in FOCEXECs in sequential files. The reason for this is that -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.

FOCUS Databases

FOCUS databases contain data written in FOCUS format. The maximum size of a FOCUS database is 2G. Through the use of partitioning, the total size can be up to 500G. See the *Describing Data* manual for information. Each database 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 database's Master File has the member name LEDGER, then the database is allocated to ddname LEDGER (if you want to override this default, you can use the USE command, a DATASET attribute in the Master File, or an Access File. All of these techniques are explained in the *Describing Data* manual.)

Allocating FOCUS Databases

FOCUS databases are formatted in 4096-byte pages. You can request both primary and secondary allocations. MVS permits up to 15 extensions of secondary space which you can request in units of tracks, cylinders, or 4096-byte blocks. We strongly recommend that you allocate FOCUS databases 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.

The following example is for a new FOCUS database:

```
//CAR   DD DSNAME=CAR.FOCUS,UNIT=SYSDA,DISP=(NEW,CTLG),
//  VOL=SER=MYVOL,SPACE=(CYL,(5,5))
```

FOCUS supplies the DCB when you issue the CREATE command which is RECFM=F, LRECL=4096, and BLKSIZE=4096. You can use the FOCUS MODIFY and FSCAN commands to maintain FOCUS databases.

If your site does not pre-allocate FOCUS databases prior to using them, you can have FOCUS automatically allocate them by issuing the following command

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

where:

`ON`

Causes FOCUS to automatically allocate FOCUS databases with a data set name of *prefix.masterfile*.FOCUS.

`OFF`

Indicates that you must allocate your FOCUS databases. OFF is the default.

Multi-Volume Support

You have the option of allocating new FOCUS databases, Fusion databases, 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 databases, Fusion databases, and FOCUS-created sequential files in the MVS 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:

- The first volume.
- Intermediate volumes.
- The 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 MVS 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 or Fusion database, 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.
	
	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 MVS Batch Environment

You have two choices for statically allocating a new multi-volume FOCUS database, Fusion database, 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 the each of the volumes to use.

type
Is the type of unit to use.

n
Is the number of units.

Reference Allocating a Multi-Volume Data Source in TSO and MVS FOCUS

In both TSO and MVS 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).

How to Allocate Specific Volumes in TSO and MVS FOCUS

Syntax

To allocate specific volumes for a multi-volume data source, use the following syntax:

- In TSO

  ```
  TSO ALLOC … VOLUME('vol1,…,voln')…
  ```

- In MVS FOCUS

  ```
  DYNAM ALLOC … VOL vol1,…,voln …
  ```

  where:

  vol1,…,voln

  Are the volume identifiers for the each of the volumes to use.

How to Specify the Number of Units in TSO and MVS FOCUS

Syntax

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 MVS 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 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 Databases

An external index is a FOCUS file that contains index, field, and segment information for one or more specified FOCUS databases. The external index is independent of its associated FOCUS database and is used to improve retrieval performance.

In MVS, the external index is automatically allocated as a temporary file when it is created using REBUILD. To create a 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
```

Disposition of FOCUS Databases

Existing FOCUS databases 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 database, 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 database relinquishes control, making the database available to the other users.

While only one TSO user or batch job is allowed to update the database, multiple users may have the database 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 database 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 database currently being modified by another user or batch job, or if the database is controlled by a FOCUS database server (sink machine), the result is a message stating that the database 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 database 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 database with DISP=OLD or NEW. This is so their addresses may be written into the database from which they are being cross-referenced. This process, called pointer resolution, does not take place if the disposition of the database is SHR or if the database resides on a FOCUS database server.

Database Security: ENCRYPT, DECRYPT and RESTRICT

The ENCRYPT Command

The syntax for the ENCRYPT command in TSO is

```
ENCRYPT FILE membername   {MASTER|FOCEXEC}
```

where *membername* is the name of the member in the PDS to be encrypted. For example, the command

```
ENCRYPT FILE EMPLOYEE
```

encrypts member EMPLOYEE of the PDS allocated to ddname MASTER.

The DECRYPT Command

The syntax for the DECRYPT command in TSO is

```
DECRYPT FILE membername   {MASTER|FOCEXEC}
```

where *membername* is the name of the member of the PDS to be decrypted.

The RESTRICT Command

The syntax of the RESTRICT command for restricting existing files is:

```
RESTRICT
ddname
ddname
.
.
.
END
```

For example:

```
RESTRICT
CAR
EMPLOYEE
END
```

FOCUS Databases and IBM Utility Programs

Although random access techniques (BDAM) are used with FOCUS databases, the files appear to the operating system as simple sequential data sets with a fixed-length, unblocked 4096-byte records. Thus FOCUS databases 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 databases 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

The FOCCOMP file contains the output from the COMPILE command

```
COMPILE focexecname
```

where:

focexecname
Is the name of the FOCEXEC.

The FOCCOMP file is used to run the compiled MODIFY procedure:

```
RUN focexecname
```

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.

Window Files

Window files contain the windows, menus, and related information created by Window Painter. There are three types of window files:

- Compiled window files are created when a Window Painter user chooses the Create a new file option, or invokes Window Painter and specifies a window file that does not exist yet. Compiled window files are also created when a window transfer file is compiled by the WINDOW COMPILE command. Compiled window files can be executed by a Dialogue Manager -WINDOW statement, and can be edited using Window Painter.
- Window transfer files are created by selecting the Create a transfer file option from the Window Painter Utilities Menu. Transfer files are uncompiled source code versions of compiled window files; they can be transferred from FOCUS running in one operating environment (for example, MVS/TSO) 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. 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.

These are described in the following sections.

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 that 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

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, TOTAL, SYSTEM 2000, MODEL 204, Millennium, Supra database files; DB2, SQL/DS, 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, ADABAS, and TOTAL data sources, you must also allocate all of the files normally used with these data sources.

Dynamically Setting the VSAM Addressing Mode

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

The syntax is:

```
{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 will not need to change the default.

TTEDIT Files

When you save a TableTalk session, a command file and a session file are saved. These files are written as follows:

- The command file is written as a member in the 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.

To edit saved TableTalk sessions, enter the command

```
TABLETALK EDIT [filename]
```

If you omit the file name, a list of all TTEDIT files will be displayed. Include a file name to edit a particular saved TableTalk session.

HOLDSTAT Files

HOLDSTAT files enable you to include FOCUS DBA information and environmental comments in HOLD and PCHOLD Master Files. Member HOLDSTAT in the distributed library ‘prefix.ERRORS.DATA’ is the default. Alternately, you may create your own HOLDSTAT or another user-specified member in your ERRORS or MASTER PDSs.

The contents of a HOLDSTAT file are 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.

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.

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 DCB attributes of the PDS are RECFM=FB, LRECL=80, and BLKSIZE a multiple of LRECL. 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 via 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

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 5-29.

HOLD Files

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.

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

SAVB Files

When you issue the command SAVB, FOCUS saves 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. The syntax is

```
SAVB [AS ddname]
```

where:

ddname
Is 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

When you issue the command SAVE, you get 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.

Temporary Master Files: HOLDMAST Files

When FOCUS HOLD files are created either under the default name HOLD, or 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 via 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 5-12 for information.

Work Files

Default allocations vary depending on the type of work file. See the *FOCUS for S/390 OS/390 and MVS Installation Guide* for details.

FOCSTACK

The 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. The maximum size of a FOCSORT file is 1G or 256K pages. 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. 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 the 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 databases. 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/REORG to a temporary file large enough to hold the data. It is recommended that you REBUILD in sections to a new file 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 database. In the LOAD phase, make sure to *add* each new section after the first. To add to a database, 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 is a sequential file, dynamically allocated, used by TableTalk.

Calling FOCUS Under TSO

You can allocate the file names required by FOCUS:

- In the logon procedure.
- Using normal batch JCL.
- Using TSO CLISTs that you execute after the logon, before FOCUS is entered.
- From within FOCUS, prior to their first use, with DYNAM ALLOCATE or TSO ALLOCATE commands. Information Builders recommends using DYNAM as its performance is superior to passing each command to TSO.

A common practice is to allocate all commonly used files in the logon procedure, and the remainder from a TSO CLIST.

Batch Operation

You can process FOCUS jobs in MVS batch mode. The MVS operator can initiate the jobs directly, or you can enter them in MVS using the TSO SUBMIT or DYNAM SUBMIT command (see *The DYNAM Command* on page 5-61 for DYNAM commands).

Note: When you design FOCUS applications for execution both online and in batch, it is important to remember that in MVS 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.

The following is a sample jobstream of MVS batch JCL:

```
//FOCUS EXEC PGM=FOCUS
//STEPLIB  DD  DSN=FOCUS.FOCLIB.LOAD,DISP=SHR
//ERRORS   DD  DSN=FOCUS.ERRORS.DATA,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//OFFLINE  DD  SYSOUT=A
//MASTER   DD  DSN=MASTER.DATA,DISP=SHR
//FOCEXEC  DD  DSN=FOCEXEC.DATA,DISP=SHR
//*     FOCUS FILE CAR
//CAR   DD  DSN=CAR.FOCUS,DISP=SHR
//SYSIN DD  *
 .
 . FOCUS commands
 .
FIN
//*
```

Each time you enter FOCUS, it automatically opens and executes the procedure PROFILE if it is present before opening SYSIN. If you operate FOCUS in batch mode, you should include a FIN statement in the PROFILE or in the SYSIN jobstream; otherwise, FOCUS will terminate with completion code 8.

Note that outside of a request or FOCEXEC, you can insert a comment line in SYSIN by starting the line with an asterisk (*).

To use an alternative PROFILE member in place of the usual one, specify the following values for the PARM parameter:

```
//FOCUS EXEC PGM=FOCUS,PARM='NOPROF'
```

or

```
//FOCUS EXEC PGM=FOCUS,PARM='PROFILE member'
```

where:

`NOPROF`
Is an optional parameter; enables you to bypass or ignore your usual PROFILE member of the FOCEXEC data set when you enter your FOCUS session.

member
Is an optional parameter; names an alternative PROFILE to execute instead of the usual PROFILE member.

Direct Entry

The FOCUS load module that controls the FOCUS environment resides in the partitioned data set FOCLIB.LOAD, member name FOCUS. The data set name may vary due to local data set naming conventions.

You can enter FOCUS either by having a logon procedure in which the load library is allocated to the ddname STEPLIB, by issuing the CALL command directly, or through a CLIST.

LOGON Procedures

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 you 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)'
```

Using CLISTs

The following is a typical FOCUS CLIST procedure that allocates the 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

FOCUS facilities under TSO include:

- FIDEL, an environment for describing full-screen data entry forms.
- TED, an optional full-screen editor for creating and editing text files.
- 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 database.

The following sections describe options in these facilities that are specific to the TSO environment.

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.

Optimizing Dialogue Manager Screen Operations Under TSO

In Dialogue Manager, the FIDEL facility -CRTFORM is used for full-screen display and update of amper variables on IBM 3270-series terminals.

The -CRTFORM statement performs a series of I/O operations to establish full-screen mode before displaying the screen and to exit full-screen mode. In some procedures, one screen follows another without any intervening non-full-screen terminal output and, in such procedures, the full-screen setup and exit sequences may be eliminated, reducing terminal I/O. The Dialogue Manager control statement -FULLSCR informs FOCUS when full-screen setup/exit operations need to be performed. The default condition is to issue the setup and exit full-screen mode for each -CRTFORM screen.

The syntax is

```
-FULLSCR   {ON|OFF}
```

where:

`ON`
Turns on full-screen I/O optimization.

`OFF`
Turns off full-screen I/O optimization.

The -FULLSCR ON statement informs FOCUS that the screen setup operation is to be performed on the next -CRTFORM statement. The setup will not be performed again nor will a full-screen mode exit be performed until the -FULLSCR OFF is executed. At that point, full-screen mode is exited and the screen is cleared.

The -FULLSCR OFF statement need not be issued. By default, FOCUS issues the exit full-screen mode statement and clears the screen when it senses that there are no lines left to be executed.

If a procedure is using consecutive -CRTFORM statements, the -FULLSCR ON statement saves three I/Os (setup, exit, and clear) for each subsequent -CRTFORM statement.

Note:

- Once the -FULLSCR ON statement has been issued, any terminal I/O other than that produced by -CRTFORM will cancel the full-screen mode. This causes an alarm to sound (if your terminal is equipped with sound). When the next -CRTFORM is executed, three asterisks (***) are displayed and the terminal waits for you to press the Enter key before it displays the next screen. This is proper execution of the Set Full-Screen Mode macro (STFSMODE) ensuring that you read messages before the screen is erased by the display of another full screen.
- You can clear the screen with -CRTCLEAR, which displays asterisks before clearing the screen. Coming from a different full-screen operation to a -CRTFORM screen (via -RUN for instance) may leave the edges of the screen either unprotected or showing unwanted characters left over from the previous screen. This is because the screen clearing I/O has been eliminated.

If one procedure executes another (via the EXEC command), and the first procedure sets -FULLSCR ON, the second procedure will behave as if the commands -FULLSCR OFF and then -FULLSCR ON were present at the top of the called procedure. This is not true for -INCLUDE procedures that behave as if they were part of the calling procedure.

The TED Editor

To enter TED under TSO, use the following syntax

```
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)
```

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 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 data set allocated to ddname FOCEXEC.
`TED X.DATA(REPT)`	Edits member REPT of data set *prefix*.X.DATA.
`TED 'USER1.X.DATA'`	Edits data set USER1.X.DATA.

You can issue TSO commands from the TED environment:

```
{TSO|MVS}    command
```

You may also substitute the MVS prefix for the TSO prefix.

You can also submit a "job" from TED by using the following syntax to submit the current file to MVS:

```
SUBmit
```

Note:

- In MVS, TED cannot edit uncataloged data sets.
- In MVS, 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.
- The use of TED on RECFM V files is not supported under TSO.
- TED may not edit files containing unprintable characters.
- In MVS, 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.

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, this is processed by TED before displaying the first screen. 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.

On TSO, since the profile is a member of FOCEXEC, it must consist of fixed-length 80-byte records.

GRAPH

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 *FOCUS for IBM Mainframe MVS/TSO 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 will appear asking for the name of the "save" file. Enter a name and press the Enter key. The screen will be 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 the 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.

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.

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 Users Manual* for more information.

Accessing 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. To access the FOCUS Menu, issue:

```
EX FMMAIN
```

The FOCUS Menu may also be invoked automatically when you enter FOCUS. To automatically invoke it, edit 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 look like:

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

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.

Accessing the FOCUS 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.

To access the FOCUS ToolKit, issue:

```
EX ISHFSHLL
```

The FOCUS ToolKit may also be invoked automatically when the user enters FOCUS. To automatically invoke it, edit the file SHELPROF FOCEXEC, which is found on either the FOCUS production disk or on any accessed disk, to read:

```
EX ISHFSHLL
-EXIT
```

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). For example, the member CAR accesses the CAR FOCUS file:

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

Accessing 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.

To access Power Reporter, issue:

```
EX PWREP
```

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.

National Language Support

FOCUS is designed with consideration for National Language Support (NLS). FOCUS stores data as entered and retrieves the data based on the current code page mapping. FOCUS can be configured for local language messages and keyboards. For installation instructions, refer to the *FOCUS for IBM Mainframe MVS/TSO Installation Guide*. Your Information Builders' representative can provide a list of available language choices.

Checking Current Language Settings

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

The syntax is:

```
? LANG
```

FOCUS supports languages such as Kanji that require two bytes of storage per character. See the *Describing Data* manual.

TSO and FOCUS Interaction

You can execute TSO commands from within FOCUS. These commands can be used in conjunction with FOCUS to create and maintain applications. You can also go directly to ISPF edit screens from within the FOCUS command environment. This means that you can start a FOCUS session and then issue TSO and ISPF commands to allocate files, create, and edit FOCUS procedures and Master Files, all directly from with FOCUS.

Additionally, you can issue command interrupts to halt processing or suppress output.

Issuing TSO Commands From Within FOCUS

You can issue almost all of the standard TSO commands from within the FOCUS environment. The exceptions are EXEC, TIME, and TSO commands that load programs, 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.

The syntax for issuing TSO commands is:

```
{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 Users Guide*. Installation-written commands should be avoided or thoroughly tested as they may or may not execute properly within FOCUS.

Executing CLISTs

It is possible to 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(0,'A1');
  -SET &COMLEN2 = HEXBYT(&COMAND.LENGTH,'A1');
  -SET &COMLEN  = &COMLEN1 || &COMLEN1 || &COMLEN1 || &COMLEN2;
  -TSO RUN ISPLINK,SELECT,&COMLEN,&COMAND
  ```

Note:

- Some commands placed in a CLIST that is invoked from within FOCUS may give unpredictable results, particularly when FOCUS has been invoked from within ISPF.
- For detailed documentation on ISPSTART or ISPLINK, consult the *System Productivity Facility (ISPF), Dialogue Management Services*, or contact your system support group.

Using TSO Commands in FOCUS Applications

Four TSO commands, COPY, LIST, LISTDS, and RENAME, are frequently used within the FOCUS environment to create and maintain FOCUS Master Files and FOCEXECs. TSO COPY and TSO LIST are program product commands, available only if you have exercised those options. From within FOCUS, you can reference data sets in the command segments either by actual data set names (as in normal TSO syntax) or by ddnames. Obviously, in most instances, the short ddname syntax will appeal simply for ease of entry. The FOCUS syntax for the ddname versions of the four TSO commands follows:

```
TSO COPY   ddname datasetname
TSO LIST   ddname
TSO LISTDS ddname
TSO RENAME ddname datasetname
```

FOCUS recognizes that you are using the ddname syntax by the absence of embedded periods (.) in the name field of the argument. When FOCUS receives one of these commands, it replaces the ddname portion of the argument with the fully qualified data set name for that ddname. FOCUS then reprints the translated command on your terminal as an informational message before sending it to TSO for execution.

For example, if you want to rename the file, GM.FOCUS (allocated as ddname CAR), to data set name CHEVY.FOCUS, you enter:

```
TSO RENAME CAR CHEVY.FOCUS
```

FOCUS responds with an informational message:

```
RENAME 'ID.GM.FOCUS' CHEVY.FOCUS
```

TSO then receives and executes the command.

Note:

- If you want to save a temporary data set you cannot use RENAME; you must use COPY, since RENAME does not catalog a temporary data set.
- In TSO commands that use more than one data set, such as copy, you can only replace the first data set name with a ddname.

If you want to suppress the printing of the informational messages you can do so by issuing the FOCUS SET MESSAGE command (in this case, SET MESSAGE=OFF).

Referring to the PDS by ddnames

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.

Referring to Concatenated Data Sets by ddnames

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 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

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 file and/or suppressing the output. To issue an interrupt, press down once on one of the following function keys:

Terminal	Function Key
IBM 327X	PA1 or ATTN
IBM 2714	ATTN
ASCII	BREAK
ASCII (full duplex)	ESC

FOCUS signals receipt of the interrupt by the message

```
FOCUS INTERRUPTED..ENTER KX,KT,RT, FX OR ?
```

and waits for the user's reply. The effect of each reply is as follows:

KX	Kill execution, stay in FOCUS.
KT	Kill typing till next terminal read.
RT	Resume typing.
FX	Kill execution, exit FOCUS.
?	Display current run-time statistics.
Other	Ignored, execution resumes.

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 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.

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.

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.

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 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 FX reply and continue to completion.

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.

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.

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

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.

The syntax for calling your procedure from within FOCUS is

```
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. The following steps detail how to accomplish this, using FOCUS facilities and the ISPF facilities known as ISPLINK and ISPEXEC:

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

With the advent of dynamic allocation techniques, it is necessary to have the ability to check on existing file allocations within the interactive environment. The FOCUS command ? TSO DDNAME provides this through three optional command formats that give you several choices regarding the information you wish to review:

`? 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.

? TSO DDNAME

This 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.

? TSO DDNAME 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
```

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.

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 a chart in this section.

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 since it is alphanumeric.

2. This statement tests if the variable &DSNAME is a blank. If it is blank (that is, 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, 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.

The following is a list of TSO ddname variables:

TSO ddname Variable	Meaning
&DDNAME	The 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, etc.
&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	The number of blocks (of BLKSIZE bytes) that can be written on to 1 track of the device.
&TRKSPERCYL	The number of tracks per cylinder for the device.
&CYLSPERDISK	The number of cylinders per disk for the device.
&BLKSWRITTEN	The total number of blocks in the data set, assuming that all blocks are BLKSIZE long.
&FOCUSPAGES	Posted only for FOCUS databases. This number is the total number of 4096-byte pages in the database. 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.

Determining 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 the 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:

Estimating 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 database 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. This is true because 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
```

The DYNAM Command

The FOCUS DYNAM command is used to manipulate data sets under MVS. 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.

The general syntax of the DYNAM command is:

```
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 MVS 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 MVS 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. It is recommended that the full keyword be used in files and truncations be used interactively.
- 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

MVS 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 MVS 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, an MVS exclusive lock is not sufficient in multi-user environments, such as FOCUS MSO. If one user were to allocate a data set as OLD, MVS 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 pre-allocated by the user) as SHR. This avoids exclusive MVS 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.

DYNAM Allocation User Exit

The DYNAM allocation user exit is an optional site-supplied routine that may be called for each data set allocation made by DYNAM. The routine may test, alter, or reject the allocation request. For more information, see Technical Memo 7860.1, *The DYNAM User Exit*.

The ALLOCATE Subcommand

The DYNAM ALLOCATE command allocates a data set. The syntax is:

```
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 specified as asterisk (*), terminal is allocated. This is used for output only. Synonyms are: DSname-DAtaset.

`[(`*`n`*`)]`
: Relative GDG number. Must be less than or equal to zero.

`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 a 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 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. 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 a 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, it is recommended to use unabbreviated keywords.

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
```

The CONCAT Subcommand

The DYNAM CONCAT command concatenates up to 16 data sets. The syntax is

```
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
```

The FREE Subcommand

The DYNAM FREE command deallocates any number of specified data sets. The syntax is

```
DYNAM FREE {DDNAME ddname [ddname...]|DSNAME dsname [dsname...]}
DYNAM FREE LONGNAME lnam
```

where:

```
DDNAME
DDN
DD
```

Required if there is no dsname; synonym is FILENAME.

```
ddname
```

Is the ddname of the data set to be freed.

```
DSNAME
DSN
DS
```

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
```

The CLOSE Subcommand

The DYNAM CLOSE command closes data sets, which cannot be freed because they are open. The syntax is

```
DYNAM CLOSE {DDNAME ddname [ddname...]|DSNAME dsname [dsname...]}
```

where:

`CLOSE`
: Can be abbreviated as CLO.

`DDNAME`
`DDN`
`DD`
: Required if there is no dsname; synonym is FILENAME.

`ddname`
: Is the ddname of the data set to be closed.

`DSNAME`
`DSN`
`DS`
: 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 *The FREE Subcommand* on page 5-72).

The COPY Subcommand

The DYNAM COPY command copies an entire MVS data set or selected PDS members. It has the following syntax:

```
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
```

The COPYDD Subcommand

The DYNAM COPYDD command copies a sequential data set, a PDS member, or a HiperFOCUS file. The syntax is:

```
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, it is recommended to use COPY instead of COPYDD.

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)
```

The DELETE Subcommand

The DYNAM DELETE command deletes an entire MVS data set or selected PDS members. The syntax to delete an entire MVS 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)
```

The RENAME Subcommand

The DYNAM RENAME command renames an entire MVS data set or selected PDS members. The syntax to rename an entire MVS 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
```

The SUBMIT Subcommand

The DYNAM SUBMIT command submits jobs to MVS. The syntax is

```
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).

The COMPRESS Subcommand

The DYNAM COMPRESS command compresses the partitioned data sets (PDS). The syntax is

```
DYNAM COMPRESS dname [dname]...
```

where:

`COMPRESS`
: Can be abbreviated as COMP.

`dname`
: Is the dsname or ddname of a PDS to be compressed. The ISPF-like lock is obtained.

If the dsname is specified, it is allocated as OLD. If the ddname is specified and status is SHR, you have to make sure that another user does not access the PDS during the compress operation.

Note: DYNAM COMPRESS uses the IBM utility IEBCOPY, and therefore can only be used when running with APF authorization.

Example Using the DYNAM COMPRESS Command

```
DYNAM COMPRESS MYDD

DYNAM COMPRESS MYID.DATA.LIB

DYNAM COMP MYDD MYID.DATA.LIB
```

Comparison of TSO Commands, JCL, and DYNAM

This section shows examples of TSO ALLOCATE and FREE commands and JCL, and the equivalent DYNAM commands.

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
```

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
```

Freeing Files

TSO:

```
TSO FREE F(FOCEXEC)
```

DYNAM:

```
DYNAM FREE FILE FOCEXEC
```

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
```

CHAPTER 6

Using FOCUS as a Client to an iWay Server

Topics:

- Client/Server Computing and Middleware
- Overview: Using FOCUS to Access Data on
- Remote Execution
- Distributed Execution

This topic describes FOCUS client/server computing, including elements of iWay Software's middleware products. An understanding of server components is essential for those planning to implement client/server applications.

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 S/390, 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.

iWay 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:

- **The Application Programming Interface (API).** This client component, built into FOCUS for S/390, requests remote services using SQL requests.
- **The 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.

In the remainder of this chapter, the term *server* refers to any iWay server.

Overview: Using FOCUS to Access Data on a Server

This topic describes processing alternatives when using FOCUS to access data on a server:

Remote Execution This approach allows you to read, analyze, consolidate, and update remote data by sending the FOCUS stack to a server.

You can also execute FOCEXECs stored on the server (called Remote Procedures), using a process known as a Remote Procedure Call (RPC).

In remote execution, the server typically handles data processing, while the client does the request generation and data presentation.

Distributed Execution In Distributed Execution (also referred to as the iWay 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 iWay, you can control:

- Whether the source code for your procedures displays in the batch output.
- Whether the 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. OFF is the default value.

Syntax How to Control USAGE Attributes in HOLD Master Files

```
SQL EDA SET USAGEFORMAT  {OFF|ON}
```

where:

OFF

Causes USAGE formats of HOLD Master Files to be defined based on the data string that is returned by the iWay 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. OFF is the default setting.

ON

USAGE formats and edit options in the HOLD Master File match those in the original Master File.

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 iWay Server for MVS—for example, setting WIDTH and PANEL parameters for reporting.

To use iWay with FOCUS for S/390, you must have the following products installed:

- Any current release of a server on any supported platform.
- FOCUS for S/390, to use as the client. Depending on your method of remote execution (described in *Remote Execution* on page 6-6 and *Distributed Execution* on page 6-14), you also need an iWay configuration file.

The iWay 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 within your session.

In VM, you issue a FILEDEF to name the configuration file. In MVS, 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

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 THE EDA SERVER
  END
```

Remote Execution

Remote execution allows users to transmit local FOCUS requests to a remote host for execution. In this operating mode, a request built on the client is shipped to the server for execution. On completion, the server returns the output to the client, which displays it in Hotscreen. 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, USERID, and PASSWORD commands at the FOCUS prompt.

  or

- Creating a FOCEXEC that issues the REMOTE DESTINATION, REMOTE USERID, and REMOTE PASSWORD commands for you.

Note: All data sources supported by servers can be accessed using remote execution.

Logging On With REMOTE Commands

To have FOCUS automatically log you on to a server, create a FOCEXEC containing the REMOTE DESTINATION, REMOTE USERID, and REMOTE PASSWORD commands.

Syntax How to Specify a Target Server: REMOTE DESTINATION Command

The REMOTE DESTINATION command directs requests to a particular server. The syntax is

```
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
*DLELIM                = 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 EDA 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

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 MVS). The syntax is

```
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)

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). The syntax is

```
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 (VM or TSO user ID) are used to sign on onto the server.

Sending Requests to a Remote Server

You can send FOCUS requests to the server for execution as follows:

- Use the REMOTE EX command from the FOCUS Session prompt to name a FOCEXEC on the client that you want to execute on the server.
- Include the commands -REMOTE BEGIN and -REMOTE END around the code you are sending to the server. Then execute the procedure as you would any other FOCUS procedure. Expansion of Dialogue Manager amper variables takes place *on the client*, before the request is forwarded to the server.
- Use TableTalk to create the request and select the Execute on Remote Server option on the final TableTalk screen.

When the server returns report output, FOCUS displays the report in Hotscreen.

Syntax How to Execute a Request Remotely Using the REMOTE EX Command

Issue the following command at the FOCUS Session prompt:

```
REMOTE EX focexecname
```

where:

`focexecname`
Is the name of a FOCEXEC stored on the client that is to be executed on the server. Note that Dialogue Manager amper variables in the FOCEXEC will be expanded *on the client* before the request is shipped to the server.

Important:

A request executed this way may not contain the commands -REMOTE BEGIN and -REMOTE END. REMOTE EX issues those commands automatically.

Example Executing a Request Remotely Using the REMOTE EX Command

The following sample FOCEXEC (EDHOURS) is executed at the FOCUS Session prompt with the REMOTE EX command.

The following code is stored on the client as FOCEXEC EDHOURS:

```
TABLE FILE EMPLOYEE
  HEADING CENTER
  "SUMMARY REPORT OF EMPLOYEE CLASSROOM HOURS"
  " "
  SUM ED_HRS BY EMP_ID
END
```

To execute this procedure on the server, issue the following command:

```
>> REMOTE EX EDHOURS
```

Using -REMOTE BEGIN and -REMOTE END to Execute Requests Remotely

Another way to execute report requests against remote data is to begin the FOCEXEC that you want executed on the server with the command -REMOTE BEGIN and follow the END command with -REMOTE END. Then execute the FOCEXEC.

Using -REMOTE BEGIN and -REMOTE END provides three advantages:

- The command for executing the procedure is the same that you would use locally.
- By adding -REMOTE BEGIN and -REMOTE END, developers can segment FOCEXECs into separate 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 within 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 within 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.

You 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 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 Hotscreen 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 FOCEXEXS
-REMOTE END
```

Example Using -INCLUDE to Call a FOCEXEC Stored on the Client

You can use the -INCLUDE command within 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 System and Error Messages

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. You can also issue the following command to display any error message:

```
? nnnn
```

where:

`nnnn`
Is the error message number.

Terminating the Remote Session: REMOTE FIN

The REMOTE FIN command logically terminates a FOCUS session with a server. It should be issued at the conclusion of remote data access. The server must be available when you issue this command.

Note: Issuing a FIN command in native FOCUS closes all active sessions.

Syntax How to Terminate a Remote Session With the REMOTE FIN Command

```
REMOTE FIN server
```

where:

`server`
Must match the server name in the configuration file.

Querying Remote Session Parameter Settings: ? REMOTE

The ? REMOTE command displays all remote session parameters in force. 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 (EDA4.3)
```

Distributed Execution

With distributed execution (also known as the iWay Data Adapter or SUFFIX=EDA), you can access all data sources accessible to a server. Your Master and Access Files tell FOCUS where to find the data.

The iWay Data Adapter provides two advantages:

- Once you set up your Master and Access Files on the client, you can use FOCUS to access remote data just as if it were local data.
- You can also join data sources across platforms. For example, join a data source on MVS 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 two 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 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 MVS, 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 SQL MVS 3.X 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 EDA 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 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 accessible through iWay 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 MVS).
- The iWay 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 ship to the DBMS. Consequently, make sure that you provide values for the ALIAS attributes in your file descriptions.
- Master Files should also contain values for ACTUAL format attributes.

Logging On to the Server With Distributed Execution

You can log on to a server by issuing SQL EDA SET commands in a FOCEXEC or at the FOCUS Session prompt.

Syntax How to Set the Server Destination With Distributed Execution

To set the server destination (an alternative to placing the server name in the Access File), use the following command:

```
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

To send a user ID and password to the server, issue

```
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 that requests will be directed to. 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 iWay 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 an MVS 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. This is because the data adapter will make one SQL call to the first data source, and then make one SQL call to the second data source for each value of the referenced field.

Issuing SQL Commands to the Server With Distributed Execution

You can issue FOCUS or SQL commands to the server. Issue the SQL commands at the FOCUS Session prompt or place them in FOCUS procedures. For example, from the FOCUS Session prompt, you could issue the following commands:

```
> sql eda
> select * from car
> end
```

Executing Stored Procedures With Distributed Execution

In addition to making data available to the client, a server can store procedures, which are called *remote procedures* or *stored procedures*.

Syntax How to Execute a Stored Procedure Using Distributed Execution

You can execute stored procedures from FOCUS, by issuing the command:

```
SQL EDA EX rpcname parm1, parm2, ...
END
```

where:

`rpcname`
Is a procedure on the server.

`parm1, parm2,...`
Are character strings sent to the server (they are the same as parameters you can pass on the execution line of a FOCEXEC).

Syntax How to Query iWay Data Adapter Settings

To view iWay 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 iWay Data Adapter also 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 iWay 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.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

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