## Contents

1 Working With Cross-Century Dates .............................................................................................................. 1-1
   The Sliding Window Technique .................................................................................................................. 1-2
      Defining a Sliding Window ...................................................................................................................... 1-3
      Creating a Dynamic Window Based on the Current Year ................................................................. 1-3
   Applying the Sliding Window Technique ................................................................................................. 1-4
      When to Supply Settings for DEFCENT and YRTHRESH ................................................................. 1-4
      Date Validation ..................................................................................................................................... 1-6
   Defining a Global Window With SET ....................................................................................................... 1-6
   Defining a Dynamic Global Window With SET ....................................................................................... 1-9
   Querying the Current Global Value of DEFCENT and YRTHRESH ...................................................... 1-12
   Defining a File-Level or Field-Level Window in a Master File .............................................................. 1-13
   Defining a Window for a Virtual Field ...................................................................................................... 1-20
   Defining a Window for a Temporary Field ........................................................................................... 1-26
   Additional Support for Cross-Century Dates ........................................................................................... 1-31

2 Managing Applications With Dialogue Manager ...................................................................................... 2-1
   Overview of Dialogue Manager Capabilities ........................................................................................... 2-2
      Overview of Dialogue Manager Variables .......................................................................................... 2-5
   Creating and Storing Procedures ............................................................................................................. 2-6
   Executing Procedures ............................................................................................................................. 2-6
      Controlling Access to Data .................................................................................................................... 2-7
   Including Comments in a Procedure ....................................................................................................... 2-7
   Overview of Dialogue Manager Commands ........................................................................................... 2-8
   Sending a Message to the User: -TYPE ................................................................................................... 2-10
   Controlling Execution: -RUN, -EXIT, and -QUIT ................................................................................ 2-11
      Executing Stacked Commands and Continuing the Procedure: -RUN .............................................. 2-11
      Executing Stacked Commands and Exiting the Procedure: -EXIT ................................................ 2-12
      Canceling Execution of the Procedure: -QUIT .................................................................................... 2-14
   Branching .................................................................................................................................................. 2-15
      -GOTO Processing ............................................................................................................................... 2-15
      Compound -IF Tests .......................................................................................................................... 2-18
      Using Operators and Functions in -IF Tests ...................................................................................... 2-19
      Screening Values With -IF Tests ........................................................................................................ 2-19
      Testing the Status of a Query ............................................................................................................... 2-23
   Looping .................................................................................................................................................... 2-24
      Ending a Loop ..................................................................................................................................... 2-26
Using Expressions:  \textbf{-SET}} 
\begin{itemize}
\item Computing a New Variable  \textbf{2-27}
\item Controlling a Loop With \textbf{-SET}  \textbf{2-28}
\item Using the DECODE Function  \textbf{2-29}
\item Using the EDIT Function  \textbf{2-30}
\item Setting a Date  \textbf{2-31}
\item Calling a Subroutine  \textbf{2-31}
\end{itemize}

Additional Facilities 
\begin{itemize}
\item Establishing Startup Conditions  \textbf{2-34}
\item Incorporating Multiple Procedures  \textbf{2-35}
\item Nesting Procedures With \textbf{-INCLUDE}  \textbf{2-38}
\item Using EXEC  \textbf{2-38}
\item Developing an Open-Ended Procedure  \textbf{2-39}
\item Debugging With \textbf{&ECHO} and \textbf{&STACK}  \textbf{2-40}
\item Locking Procedure Users Out of FOCUS  \textbf{2-41}
\item Writing to Files:  \textbf{-WRITE}  \textbf{2-42}
\end{itemize}

Using Variables in Procedures  
\begin{itemize}
\item Querying the Values of Variables  \textbf{2-48}
\item Local Variables  \textbf{2-49}
\item Global Variables  \textbf{2-50}
\item System Variables  \textbf{2-51}
\item Statistical Variables  \textbf{2-57}
\item Special Variables  \textbf{2-59}
\item Using Variables to Alter Commands  \textbf{2-60}
\item Evaluating a Variable Immediately  \textbf{2-60}
\item Concatenating Variables  \textbf{2-62}
\end{itemize}

Supplying Values for Variables at Run Time  \textbf{2-63}
\begin{itemize}
\item Supplying Values Without Prompting  \textbf{2-65}
\item Supplying Values With \textbf{-DEFAULTS}  \textbf{2-67}
\item Supplying Values With \textbf{-SET}  \textbf{2-68}
\item Supplying Values With \textbf{-READ}  \textbf{2-69}
\item Direct Prompting With \textbf{-PROMPT}  \textbf{2-70}
\item Full-Screen Data Entry With \textbf{-CRTFORM}  \textbf{2-72}
\item Selecting Data From Menus and Windows With \textbf{-WINDOW}  \textbf{2-72}
\item Implied Prompting  \textbf{2-72}
\item Verifying Input Values  \textbf{2-73}
\end{itemize}

Dialogue Manager Quick Reference  \textbf{2-77}
\begin{itemize}
\item System Defaults and Limits  \textbf{2-108}
\end{itemize}
Contents

3 Designing Windows With Window Painter ................................................................. 3-1
   Introduction .............................................................................................................. 3-2
   How Do Window Applications Work? ....................................................................... 3-2
   Window Files and Windows ..................................................................................... 3-3
   Types of Windows You Can Create ......................................................................... 3-4
   Creating Windows .................................................................................................... 3-11
   Integrating Windows and the FOCEXEC ................................................................. 3-19
   Transferring Control in Window Applications ..................................................... 3-20
   Return Values .......................................................................................................... 3-22
   Goto Values ............................................................................................................. 3-23
   Window System Variables ...................................................................................... 3-24
   Testing Function Key Values ................................................................................. 3-24
   Executing a Window From the FOCUS Prompt .................................................... 3-26
   Tutorial: A Menu-Driven Application ..................................................................... 3-27
   Creating the Application FOCEXEC ....................................................................... 3-28
   Creating the Window File ....................................................................................... 3-31
   Executing the Application ....................................................................................... 3-47
   Window Painter Screens .......................................................................................... 3-47
   Invoking Window Painter ....................................................................................... 3-48
   Entry Menu ............................................................................................................. 3-49
   Main Menu ............................................................................................................. 3-50
   Window Creation Menu ........................................................................................... 3-52
   Window Design Screen ........................................................................................... 3-54
   Window Options Menu ............................................................................................ 3-56
   Utilities Menu ......................................................................................................... 3-67
   Transferring Window Files ....................................................................................... 3-70
   Creating a Transfer File ........................................................................................... 3-70
   Transferring the File to the New Environment .................................................... 3-71
   Editing the Transfer File .......................................................................................... 3-71
   Compiling the Transfer File ..................................................................................... 3-77

4 Setting Parameters: SET ......................................................................................... 4-1

5 Defining Word Substitutions: LET .......................................................................... 5-1
   LET Command Syntax ............................................................................................ 5-3
   Variable Substitutions in LET Phrases .................................................................... 5-5
   Using LET Phrases in COMPUTE and DEFINE Statements .................................. 5-7
   Null Substitutions .................................................................................................. 5-7
   Multiple-Line Substitutions .................................................................................... 5-8
   Recursive Substitutions .......................................................................................... 5-8
   Checking Current LET Substitutions: ? LET ....................................................... 5-9
   Clearing LET Substitutions: LET CLEAR ............................................................. 5-11
   Saving LET Equivalences in a File: LET SAVE ..................................................... 5-12
   Defining PF Keys ..................................................................................................... 5-12

Developing Applications
<table>
<thead>
<tr>
<th>Master Files and Diagrams</th>
<th>A-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-3</td>
<td>The EMPLOYEE Database</td>
</tr>
<tr>
<td>A-4</td>
<td>The EMPLOYEE Master File</td>
</tr>
<tr>
<td>A-5</td>
<td>The EMPLOYEE Structure Diagram</td>
</tr>
<tr>
<td>A-6</td>
<td>The JOBFILE Database</td>
</tr>
<tr>
<td>A-7</td>
<td>The JOBFILE Master File</td>
</tr>
<tr>
<td>A-8</td>
<td>The JOBFILE Structure Diagram</td>
</tr>
<tr>
<td>A-7</td>
<td>The EDUCFILE Database</td>
</tr>
<tr>
<td>A-8</td>
<td>The EDUCFILE Master File</td>
</tr>
<tr>
<td>A-9</td>
<td>The EDUCFILE Structure Diagram</td>
</tr>
<tr>
<td>A-10</td>
<td>The PROD Database</td>
</tr>
<tr>
<td>A-10</td>
<td>The PROD Master File</td>
</tr>
<tr>
<td>A-11</td>
<td>The PROD Structure Diagram</td>
</tr>
<tr>
<td>A-11</td>
<td>The CAR Database</td>
</tr>
<tr>
<td>A-11</td>
<td>The CAR Master File</td>
</tr>
<tr>
<td>A-12</td>
<td>The CAR Structure Diagram</td>
</tr>
<tr>
<td>A-13</td>
<td>The LEDGER Database</td>
</tr>
<tr>
<td>A-13</td>
<td>The LEDGER Master File</td>
</tr>
<tr>
<td>A-13</td>
<td>The LEDGER Structure Diagram</td>
</tr>
<tr>
<td>A-14</td>
<td>The FINANCE Database</td>
</tr>
<tr>
<td>A-14</td>
<td>The FINANCE Master File</td>
</tr>
<tr>
<td>A-14</td>
<td>The FINANCE Structure Diagram</td>
</tr>
<tr>
<td>A-15</td>
<td>The REGION Database</td>
</tr>
<tr>
<td>A-15</td>
<td>The REGION Master File</td>
</tr>
<tr>
<td>A-15</td>
<td>The REGION Structure Diagram</td>
</tr>
<tr>
<td>A-16</td>
<td>The COURSES Database</td>
</tr>
<tr>
<td>A-16</td>
<td>The COURSES Master File</td>
</tr>
<tr>
<td>A-16</td>
<td>The COURSES Structure Diagram</td>
</tr>
<tr>
<td>A-17</td>
<td>The EMPDATA Database</td>
</tr>
<tr>
<td>A-17</td>
<td>The EMPDATA Master File</td>
</tr>
<tr>
<td>A-17</td>
<td>The EMPDATA Structure Diagram</td>
</tr>
<tr>
<td>A-18</td>
<td>The EXPERSON Database</td>
</tr>
<tr>
<td>A-18</td>
<td>The EXPERSON Master File</td>
</tr>
<tr>
<td>A-18</td>
<td>The EXPERSON Structure Diagram</td>
</tr>
<tr>
<td>A-19</td>
<td>The TRAINING Database</td>
</tr>
<tr>
<td>A-19</td>
<td>The TRAINING Master File</td>
</tr>
<tr>
<td>A-19</td>
<td>The TRAINING Structure Diagram</td>
</tr>
<tr>
<td>A-20</td>
<td>The PAYHIST File</td>
</tr>
<tr>
<td>A-20</td>
<td>The PAYHIST Master File</td>
</tr>
</tbody>
</table>
Preface

This documentation set describes how to use FOCUS® Version 7.0. The documentation set consists of the following components:

- 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 *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 environment.

The users’ documentation for FOCUS Version 7.0 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.

References to MVS apply to all supported versions of the OS/390 and MVS operating environments.
Audience

This manual is for any FOCUS user.

How This Manual Is Organized

This manual includes the following chapters:

<table>
<thead>
<tr>
<th>Chapter/Appendix</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Working With Cross-Century Dates</td>
<td>Describes techniques for assigning a century to dates stored with two-digit years.</td>
</tr>
<tr>
<td>2 Managing Applications With Dialogue Manager</td>
<td>Describes how to make report procedures more dynamic by using Dialogue Manager control statements and variables.</td>
</tr>
<tr>
<td>3 Designing Windows With Window Painter</td>
<td>Describes how to create FOCUS windows and menus that work in conjunction with a FOCEXEC.</td>
</tr>
<tr>
<td>4 Setting Parameters: SET</td>
<td>Lists commands you use to control output, work areas, and many other FOCUS features.</td>
</tr>
<tr>
<td>5 Defining Word Substitutions: LET</td>
<td>Describes how to define string substitutions that can be used in FOCUS report requests.</td>
</tr>
<tr>
<td>6 FOCUS Query Commands</td>
<td>Describes how to use query commands to retrieve information about the FOCUS environment.</td>
</tr>
<tr>
<td>7 Enhancing Application Performance</td>
<td>Describes FOCUS facilities for increasing the speed of your application.</td>
</tr>
<tr>
<td>8 Euro Currency Support</td>
<td>Describes how to perform currency conversions according to the rules established by the European Union.</td>
</tr>
<tr>
<td>A Master Files and Diagrams</td>
<td>Contains Master Files and diagrams of sample data sources used in the documentation examples.</td>
</tr>
<tr>
<td>B Error Messages</td>
<td>Describes how to write subroutines that can be called from FOCUS.</td>
</tr>
</tbody>
</table>
### Summary of New Features

The FOCUS features and enhancements described in this manual are listed in the following table:

<table>
<thead>
<tr>
<th>New Feature</th>
<th>Version/Release</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMLE, controlling IBM’s Dynamic Language Environment for</td>
<td>7.0.8R</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>IBI-supplied and user-written subroutines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Dialogue Manager variables:</td>
<td>7.0.8R</td>
<td>Chapter 2, Managing Applications With Dialogue Manager</td>
</tr>
<tr>
<td>&amp;YYMD, &amp;MDYY, &amp;DMYY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TESTDATE, temporarily altering your FOCUS system date for a given</td>
<td>7.0.8R</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>application program</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATEDISPLAY, displaying base dates in a FOCUS report</td>
<td>7.0.8R</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>YRTHRESH as an offset from the current year for a moving century window</td>
<td>7.0.8R</td>
<td>Chapter 4, Setting Parameters: SET and Chapter 1, Working With</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cross-Century Dates.</td>
</tr>
<tr>
<td>Querying which PTFs have been applied for a specific release</td>
<td>7.0.8</td>
<td>Chapter 6, FOCUS Query Commands</td>
</tr>
<tr>
<td>Assigning screening conditions to a file for reporting purposes</td>
<td>7.0.8</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>Date handling for the year 2000</td>
<td>7.0.8</td>
<td>Chapter 1, Working With Cross-Century Dates</td>
</tr>
<tr>
<td>Estimating SORTWORK sizes for external sorts</td>
<td>7.0.6</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>PRINTPLUS, enhancements to FOCUS Report Writer display options</td>
<td>7.0.6</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>Cross-century dates in FOCUS applications</td>
<td>7.0.6</td>
<td>Chapter 1, Working With Cross-Century Dates</td>
</tr>
</tbody>
</table>

**Developing Applications**
<table>
<thead>
<tr>
<th>New Feature</th>
<th>Version/Release</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project 2000: Cross-century dates in FOCUS applications</td>
<td>7.0.5</td>
<td>Chapter 1, Working With Cross-Century Dates</td>
</tr>
<tr>
<td>MINIO, a new I/O buffering technique that improves performance (MVS only)</td>
<td>7.0.5</td>
<td>Chapter 4, Setting Parameters: SET and Chapter 7, Enhancing Application Performance</td>
</tr>
<tr>
<td>DATETIME, setting date and time stamp in reports</td>
<td>7.0.5</td>
<td>Chapter 4, Setting Parameters: SET</td>
</tr>
<tr>
<td>Checking current language settings with ? Lang</td>
<td>7.0.1</td>
<td>Chapter 6, FOCUS Query Commands</td>
</tr>
</tbody>
</table>

**Documentation Conventions**

The following conventions apply throughout this manual:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THIS TYPEFACE</td>
<td>Denotes a command that you must enter in uppercase, exactly as shown.</td>
</tr>
<tr>
<td>this typeface</td>
<td>Denotes a value that you must supply.</td>
</tr>
<tr>
<td>( )</td>
<td>Indicates two choices. You must type one of these choices, not the braces.</td>
</tr>
<tr>
<td></td>
<td>Separates two mutually exclusive choices in a syntax line. Type one of these choices, not the symbol.</td>
</tr>
<tr>
<td>( )</td>
<td>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.</td>
</tr>
<tr>
<td>underscore</td>
<td>Indicates the default value.</td>
</tr>
<tr>
<td>...</td>
<td>Indicates that you can enter a parameter multiple times. Type only the information, not the ellipsis points.</td>
</tr>
<tr>
<td>. .</td>
<td>Indicates that there are (or could be) intervening or additional commands.</td>
</tr>
</tbody>
</table>
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.ibi.com, to view a current listing of our publications and to place an order.

Information Builders Systems Journal

The Information Builders Systems Journal is a unique technical publication dedicated to providing you with the latest information necessary to enhance your use of FOCUS and all other Information Builders products.

Through its detailed articles, illustrated with code, screen shots, and other visuals, the Journal challenges you to develop better reporting habits, customize features to enhance your systems applications, utilize its tips and techniques for better performance and productivity, and so much more.


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.ibi.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.ibi.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, or call (800) 969-INFO.
Information You Should Have

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

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

- The exact nature of the problem:
  - Are the results or the format incorrect; are the text or calculations missing or misplaced?
  - The error message and code, if applicable.
  - Is this related to any other problem?

- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?

- What release of the operating system are you using? Has it, FOCUS, your security system, or an interface system changed?

- Is this problem reproducible? If so, how?

- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two 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.ibi.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.ibi.com) or call (800) 969-INFO to speak to an Education Representative.
1 Working With Cross-Century Dates

Topics:
- The Sliding Window Technique
- Applying the Sliding Window Technique
- Defining a Global Window With SET
- Defining a Dynamic Global Window With SET
- Querying the Current Global Value of DEFCENT and YRTHRESH
- Defining a File-Level or Field-Level Window in a Master File
- Defining a Window for a Virtual Field
- Defining a Window for a Temporary Field
- Additional Support for Cross-Century Dates

Many existing business applications use two digits to designate a year, instead of four digits. When they receive a value for a year, such as 00, they typically interpret it as 1900, assuming that the first two digits are 19, for the twentieth century. These applications require a way to handle dates when the century changes (for example, from the twentieth to the twenty-first), or when they need to perform comparisons or arithmetic on dates that span more than one century.

The cross-century date feature described in this topic enables the correct interpretation of the century if it is not explicitly provided or assumed to be the twentieth. The feature is application-based, that is, it involves modifications to procedures or metadata so that dates are accurately interpreted and processed. The feature is called the sliding window technique.

When Do You Use the Sliding Window Technique?

If your application accesses dates that contain an explicit century, the century is accepted as is. Your application can run correctly across centuries, and you do not need to use the sliding window technique.

If your application accesses dates without explicit centuries, they assume the default value 19. Your application will require remediation, such as the sliding window technique, to ensure the correct interpretation of the century if the default is not valid, and to run as expected in the next century.

Details on when to use the sliding window technique are provided later in this topic.

This topic does not cover remediation options such as date expansion, which requires that data be changed in the data source to accommodate explicit century values. For a list of Information Builders documentation on remediation, see your latest Publications Catalog.
Working With Cross-Century Dates

This topic covers the use of the sliding window technique in reporting applications. It also includes reference information on the use of the technique with FOCUS MODIFY requests. For additional information on implementing this technique with Maintain, see your database maintenance documentation.

The Sliding Window Technique

With the sliding window technique, you do not need to change stored data from a 2-digit year format to a 4-digit year format in order to determine the century. Instead, you can continue storing 2-digit years and expand them when your application accesses them.

The sliding window technique recognizes that the earliest and latest values for a single date field in most business applications are within 100 years of one another. For example, a human resources application typically contains a field for the birth date of each active employee. The difference in the birth date (or age) of the oldest active employee and the youngest active employee is not likely to be more than 100.

The technique is implemented as follows:

- You define the start of a 100-year sliding window by supplying two values: one for the default century (DEFCENT) and one for the year threshold (YRTHRESH). For example, a value of 19 for the century, combined with a value of 60 for the threshold, creates a window that starts in 1960 and ends in 2059.

- The threshold provides a way to assign a value to the century of a 2-digit year:
  - A year greater than or equal to the threshold assumes the value of the default century (DEFCENT). Using the sample value 19 for the default century and 60 for the threshold, a 2-digit year of 70 is interpreted as 1970 (70 is greater than 60).
  - A year less than the threshold assumes the value of the default century plus 1 (DEFCENT + 1). Using the same sample values (19 and 60), a 2-digit year of 50 is interpreted as 2050 (50 is less than 60), and a 2-digit year of 00 is interpreted as 2000 (00 is also less than 60).

The conversion rule for this example is illustrated as follows:

<table>
<thead>
<tr>
<th>0 &lt; YRTHRESH = 60 ≥ 99</th>
</tr>
</thead>
<tbody>
<tr>
<td>⌈</td>
</tr>
<tr>
<td>⌊</td>
</tr>
<tr>
<td>Century = DEFCENT + 1</td>
</tr>
<tr>
<td>(20)</td>
</tr>
<tr>
<td>Century = DEFCENT (19)</td>
</tr>
</tbody>
</table>

Any 2-digit year is assumed to fall within the window. You must handle dates that fall outside the defined window by coding.

Each file or each date field used in an application can have its own conversion rule, which provides the flexibility required by most applications.
Defining a Sliding Window

You can define a sliding window in several ways, depending on the specific requirements of your application:

• **Globally.** The SET DEFCENT and SET YRTHRESH commands define a window on a global level.

• **On a file level.** The FDEFCENT and FYRTHRESH attributes in a Master File define a window on a file level, allowing the correct interpretation of date fields from multiple files that span different time periods.

• **On a field level.** The DEFCENT and YRTHRESH attributes in a Master File define a window on a field level, allowing the correct interpretation of date fields, within a single file, that span different time periods.

• **For a virtual field.** The DEFCENT and YRTHRESH parameters on a DEFINE command, in either a request or a Master File, define a window for a virtual field.

• **For a temporary field.** The DEFCENT and YRTHRESH parameters on a COMPUTE command define a window for a temporary field.

If you define more than one window using any of the preceding methods, the precedence is as follows:

1. DEFCENT and YRTHRESH on a DEFINE or COMPUTE command.
2. DEFCENT and YRTHRESH field-level attributes in a Master File.
3. FDEFCENT and FYRTHRESH file-level attributes in a Master File.
4. SET DEFCENT and SET YRTHRESH on a global level; if you do not specify values, the defaults are used (DEFCENT = 19, YRTHRESH = 0).

Creating a Dynamic Window Based on the Current Year

An optional feature of the sliding window technique enables you to create a dynamic window, defining the start of a 100-year span based on the current year. The start year and threshold for the window automatically change at the beginning of each new year.

If an application requires that a window’s start year change when a new year begins, use of this feature avoids the necessity of manually re-coding it.

To implement this feature, YRTHRESH or FYRTHRESH is offset from the current year, or given a negative value.

For example, if the current year is 1999 and YRTHRESH is set to -38, a window from 1961 to 2060 is created. The start year 1961 is derived by subtracting 38 (the value of YRTHRESH) from 1999 (the current year). To interpret dates that fall within this window, the threshold 61 is used.
Working With Cross-Century Dates

At the beginning of the year 2000, a new window from 1962 to 2061 is automatically created; for dates that fall within this window, the threshold 62 is used. In the year 2001, the window becomes 1963 to 2062, and the threshold is 63, and so on.

With each new year, the start year for the window is incremented by one.

When using this feature, do not code a value for DEFCENT or FDEFCENT, since the feature is designed to automatically calculate the value for the default century. Be aware of the following:

- If you do code a value for DEFCENT on the field level in a Master File, or for FDEFCENT on the file level in a Master File, the feature will not work as intended. The value for the century, which is automatically calculated by YRTHRESH by design, will be reset to the value you code for DEFCENT or FDEFCENT.

- If you code a value for DEFCENT anywhere other than the field level in a Master File (for example, on the global level), and YRTHRESH is negative, the coded value will be ignored. The default century will be automatically calculated as designed.

Applying the Sliding Window Technique

To apply the sliding window technique correctly, you need to understand the difference between a date format (formerly called a smart date) and a legacy date:

- A date format refers to an internally stored integer that represents the number of days between a real date value and a base date (either December 31, 1900, for dates with YMD or YYMD format; or January 1901, for dates with YM, YYM, YQ, or YYQ format). A Master File does not specify a data type or length for a date format; instead, it specifies display options such as D (day), M (month), Y (2-digit year), or YY (4-digit year). For example, MDYY in the FORMAT (also known as USAGE) attribute of a Master File is a date format. A real date value such as March 5, 1999, displays as 03/05/1999, and is internally stored as the offset from December 31, 1900.

- A legacy date refers to an integer, packed decimal, double precision, floating point, or alphanumeric format with date edit options, such as I6YMD, A6MDY, I8YYMD, or A8MDYY. For example, A6MDY is a 6-byte alphanumeric string; the suffix MDY indicates how Information Builders will return the data in the field. The sample value 030599 displays as 03/05/99.

For details on date fields, see your documentation on describing data.
When to Supply Settings for DEFCENT and YRTHRESH

The rest of this topic refers simply to DEFCENT when either DEFCENT or FDEFCENT applies, and to YRTHRESH when either YRTHRESH or FYRTHRESH applies.

Supply settings for DEFCENT and YRTHRESH in the following cases:

- When you issue a DEFINE or COMPUTE command to convert a legacy date without century digits to a date format with century digits (for example, to convert the format I6YMD to YYMD). With DEFINE and COMPUTE, DEFCENT and YRTHRESH do not work directly on legacy dates; for example, you cannot use them to convert the legacy date format I6YMD to the legacy date format I8YYMD.

- When a DEFINE command, COMPUTE command, or Dialogue Manager -SET statement calls a function or subroutine, supplied by Information Builders, that uses legacy dates, and the input date does not contain century digits.

  On input, the subroutine will use the window defined for an I6 legacy date field (with edit options). The output format may be I8 (again, with edit options), which includes a 4-digit year.

- When data is entered or changed in a date format field in a FOCUS data source, or an SQL date is entered or changed in a Relational Database Management System (RDBMS), and the input date does not contain century digits.

  For example, you can use the sliding window technique in applications that use FIXFORM or CRTFORM with MODIFY.

- When a data source is read, and the ACTUAL attribute in the Master File is non-date specific (for example, A6, I6, or P6), without century digits, and the FORMAT or USAGE attribute specifies a date format. This case does not apply to FOCUS data sources.

Follow these rules when implementing the sliding window technique:

- Specify values for both DEFCENT and YRTHRESH to ensure consistent coding and accurate results, except when YRTHRESH has a negative value. In that case, specify a value for YRTHRESH only; do not code a value for DEFCENT.

- Do not use DEFCENT and YRTHRESH with ON TABLE SET.

Finally, keep in mind that the sliding window technique does not change the way existing data is stored. Rather, it accurately interprets data during application processing.
Working With Cross-Century Dates

**Reference**

**Restrictions With MODIFY**

The following results occur when you use the sliding window technique with a MODIFY request or FOCCOMP procedure:

- A MODIFY request compiled prior to Version 7.0 Release 6, when run with global SET DEFCENT and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1886 error message. You must recompile the MODIFY request.

- A MODIFY request compiled in Version 7.0 Release 6, when run with global SET DEFCENT and SET YRTHRESH settings, or with file-level or field-level settings, yields a FOC1885 warning message.

- A FOCCOMP procedure, compiled with global SET DEFCENT and SET YRTHRESH settings, and run in releases prior to Version 7.0 Release 6, yields a FOC548 invalid version message. You must recompile the MODIFY request.

- A FOCCOMP procedure that contains DEFCENT/YRTHRESH or FDEFCENT/FYRTHRESH attributes in the associated Master File, and run in releases prior to Version 7.0 Release 6, yields a FOC306 description error message.

**Date Validation**

Date formats are validated on input. For example, 11/99/1999 is rejected as input to a date field formatted as MDYY, because 99 is not a valid day. Information Builders generates an error message.

Legacy dates are not validated. The date 11991999, described with the format A8MDYY, is accepted, even though it, too, contains the invalid day 99.

**Defining a Global Window With SET**

The SET DEFCENT and SET YRTHRESH commands define a window on a global level. The time span created by the SET commands applies to every 2-digit year used by the application unless you specify file-level or field-level windows elsewhere.

For details on specifying parameters that govern the environment, see your documentation on the SET command.
Defining a Global Window With SET

Syntax

**How to Define a Global Window With SET**

To define a global window, issue two SET commands.

The first command is

```
SET DEFCENT = {cc | 19}
```

where:

`cc`

Is the century for the start date of the window. If you do not supply a value, `cc` defaults to 19, for the twentieth century.

The second command is

```
SET YRTHRESH = {[-] yy | 0}
```

where:

`yy`

Is the year threshold for the window. If you do not supply a value, `yy` defaults to zero (0).

If `yy` is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If `yy` is a negative number (-`yy`), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example

**Defining a Global Window With SET**

In the following request, the SET command defines a global window from 1983 to 2082.

As SET syntax allows, the command is entered on one line, with the parameters separated by a comma. You do not need to repeat the keyword SET for YRTHRESH.

The DEFINE command converts the legacy date EFFECT_DATE into the date format NEW_DATE. It creates NEW_DATE as a virtual field, derived from the existing field EFFECT_DATE. The format of EFFECT_DATE is I6YMD, which is a 2-digit year. NEW_DATE is formatted as YYMD, which is a 4-digit year. For details on DEFINE, see your documentation on creating reports.

The request is:

```
SET DEFCENT = 19, YRTHRESH = 83
DEFINE FILE EMPLOYEE
NEW_DATE/YYMD = EFFECT_DATE;
END

TABLE FILE EMPLOYEE
PRINT EFFECT_DATE NEW_DATE BY EMP_ID
END
```
Working With Cross-Century Dates

In the report, the value of the 2-digit year 82 is less than the threshold 83, so it assumes the value 20 for the century (DEFCENT + 1) and is returned as 2082 in the NEW_DATE column. The other year values (83 and 84) are greater than or equal to the threshold 83, so their century defaults to the value 19 (DEFCENT); they are returned as 1983 and 1984 under NEW_DATE.

The report is:

```
PAGE 1

EMP_ID  EFFECT_DATE  NEW_DATE
-------  -----------  --------
071382660  82/11/01  2082/11/01
112847612                1900/12/31
117593129  82/11/01  2082/11/01
119265415                1900/12/31
119329144  83/01/01  1983/01/01
123764317  83/03/01  1983/03/01
126724188                1900/12/31
219984371                1900/12/31
326179357  82/12/01  2082/12/01
451123478  84/09/01  1984/09/01
543729165                1900/12/31
818692173  83/05/01  1983/05/01
```

In the example, missing date values appear as blanks by default. To retrieve the base date value for the NEW_DATE field instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request. The base date value for NEW_DATE, which is formatted as YYMD, is returned as 1900/12/31:

```
PAGE 1

EMP_ID  EFFECT_DATE  NEW_DATE
-------  -----------  --------
071382660                1900/12/31
112847612                1900/12/31
117593129  82/11/01  2082/11/01
119265415                1900/12/31
119329144  83/01/01  1983/01/01
123764317  83/03/01  1983/03/01
126724188                1900/12/31
219984371                1900/12/31
326179357  82/12/01  2082/12/01
451123478  84/09/01  1984/09/01
543729165                1900/12/31
818692173  83/05/01  1983/05/01
```

If NEW_DATE had a YYM format, the base date would appear as 1901/01. If it had a YYQ format, it would appear as 1901 Q1.
Defining a Dynamic Global Window With SET

If the value of NEW_DATE is 0 and SET DATEDISPLAY = OFF (the default), blanks are displayed. With SET DATEDISPLAY = ON, the base date is displayed instead of blanks. Zero (0) is treated as an offset from the base date, which results in the base date.

For details on SET DATEDISPLAY, see your documentation on the SET command.

Defining a Dynamic Global Window With SET

This topic illustrates the creation of a dynamic window using the global command SET YRTHRESH. You can also implement this feature on the file and field level, and on a DEFINE or COMPUTE.

With this option of the sliding window technique, the start year and threshold for the window automatically change at the beginning of each new year. The default century (DEFCENT) is automatically calculated.

You can use SET TESTDATE to alter the system date when testing a dynamic window (that is, when YRTHRESH has a negative value). However, when testing a dynamic window defined in a Master File, you must issue a CHECK FILE command each time you issue a SET TESTDATE command. CHECK FILE reloads the Master File into memory and ensures the correct recalculation of the start date of the dynamic window. For details on SET TESTDATE, see your documentation on the SET command. For details on CHECK FILE, see your documentation on describing data.

Example Defining a Dynamic Global Window With SET

In the following request, the COMPUTE command calls the subroutine AYMD, supplied by Information Builders. AYMD adds one day to the input field, HIRE_DATE; the output field, HIRE_DATE_PLUS_ONE, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. HIRE_DATE_PLUS_ONE is formatted as I8YYMD, which is a legacy date with a 4-digit year.

The subroutine uses the YRTHRESH value set at the beginning of the request to create a dynamic window for the input field HIRE_DATE. The start date of the window is incremented by one at the beginning of each new year. Notice that DEFCENT is not coded, since the default century is automatically calculated whenever YRTHRESH has a negative value.

The subroutine inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

Sample values are shown in the reports for 1999, 2000, and 2018, which follow the request.
Working With Cross-Century Dates

For details on AYMD, see your documentation on creating reports.

The request is:

SET YRTHRESH = -18

TABLE FILE EMPLOYEE
PRINT HIRE_DATE AND COMPUTE
  HIRE_DATE_PLUS_ONE/I8YMD = AYMD(HIRE_DATE, 1, HIRE_DATE_PLUS_ONE);
END

In 1999, the window spans the years 1981 to 2080. The threshold is 81 (1999 - 18). In the report, the 2-digit year 80 is less than the threshold 81, so it assumes the value 20 for the century (DEFCENT + 1), and is returned as 2080 in the HIRE_DATE_PLUS_ONE column. The other year values (81 and 82) are greater than or equal to the threshold 81, so their century defaults to the value of DEFCENT (19); they are returned as 1981 and 1982.

The report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRE_DATE</td>
<td>HIRE_DATE_PLUS_ONE</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/06/03</td>
</tr>
<tr>
<td>81/07/01</td>
<td>1981/07/02</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/05</td>
</tr>
<tr>
<td>82/08/01</td>
<td>1982/08/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/05</td>
</tr>
<tr>
<td>82/07/01</td>
<td>1982/07/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>1981/07/02</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/02</td>
</tr>
<tr>
<td>82/02/02</td>
<td>1982/02/03</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/02</td>
</tr>
<tr>
<td>81/11/02</td>
<td>1981/11/03</td>
</tr>
</tbody>
</table>
Defining a Dynamic Global Window With SET

In 2000, the window spans the years 1982 to 2081. The threshold is 82 (2000 - 18). In the report, the 2-digit years 80 and 81 are less than the threshold; for the century, they assume the value 20 (DEFCENT + 1). The 2-digit year 82 is equal to the threshold; for the century, it defaults to the value 19 (DEFCENT).

The report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HIRE_DATE</th>
<th>HIRE_DATE_PLUS_ONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/06/02</td>
<td>2080/06/03</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/02</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/05</td>
</tr>
<tr>
<td>82/08/01</td>
<td>1982/08/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/05</td>
</tr>
<tr>
<td>82/07/01</td>
<td>1982/07/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/02</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/02</td>
</tr>
<tr>
<td>82/02/02</td>
<td>1982/02/03</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/02</td>
</tr>
<tr>
<td>81/11/02</td>
<td>2081/11/03</td>
</tr>
</tbody>
</table>

Running the report in 2018 illustrates the automatic recalculation of DEFCENT from 19 to 20. In 2018, the window spans the years 2000 to 2099. The threshold is 0 (2018 - 18). A 2-digit year greater than or equal to 0 defaults to the recalculated value 20 (DEFCENT).

Since all the values for the HIRE_DATE year are greater than 0, their century defaults to 20.

The report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>HIRE_DATE</th>
<th>HIRE_DATE_PLUS_ONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/06/02</td>
<td>2080/06/03</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/02</td>
</tr>
<tr>
<td>82/05/01</td>
<td>2082/05/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>2082/01/05</td>
</tr>
<tr>
<td>82/08/01</td>
<td>2082/08/02</td>
</tr>
<tr>
<td>82/01/04</td>
<td>2082/01/05</td>
</tr>
<tr>
<td>82/07/01</td>
<td>2082/07/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/02</td>
</tr>
<tr>
<td>82/04/01</td>
<td>2082/04/02</td>
</tr>
<tr>
<td>82/02/02</td>
<td>2082/02/03</td>
</tr>
<tr>
<td>82/04/01</td>
<td>2082/04/02</td>
</tr>
<tr>
<td>81/11/02</td>
<td>2081/11/03</td>
</tr>
</tbody>
</table>
Querying the Current Global Value of DEFCENT and YRTHRESH

You can query the current global value of DEFCENT and YRTHRESH.

**Syntax**  
How to Query the Current Global Value of DEFCENT and YRTHRESH

The syntax is

```
? SET [ALL]
```

where:

**ALL**

Returns values for every possible environment setting. Excluding it generates a shorter list of the most common settings.

**Example**  
Querying the Recalculated Value of DEFCENT

Enter

```
? SET
```

to query the current global value of DEFCENT as recalculated in the year 2018, using the request in the previous example (defining a dynamic global window with SET).

The following partial response shows the new value of DEFCENT, as well as the value of YRTHRESH:

```
PARAMETER SETTINGS

ALL       OFF       HDAY       .       PRINT       ONLINE
.         .         .         .         .         .
.         .         .         .         .         .
.         .         .         .         .         .
.         .         .         .         .         .
DEFCENT   20        PAGE-NUM   ON       TRACKIO     ON
.         .         .         .         .         .
.         .         .         .         .         .
.         .         .         .         .         .
FOCSTACK SIZE 8   PREFIX     .       YRTHRESH   0
```
Defining a File-Level or Field-Level Window in a Master File

In this implementation of the sliding window technique, you change the metadata used by an application. Two pairs of Master File attributes enable you to define a window on a file or field level:

- The FDEFCENT and FYRTHRESH attributes define a window on a file level. They enable the correct interpretation of legacy date fields from multiple files that span different time periods.
  
  A file-level window takes precedence over a global window for the dates associated with that file.

- The DEFCENT and YRTHRESH attributes define a window on a field level, enabling the correct interpretation of legacy date fields, within a single file, that span different time periods. Each legacy date field in a file can have its own window. For example, in an insurance application, the range of dates for date of birth may be from 1910 to 2009, and the range of dates for expected death may be from 1990 to 2089.
  
  A field-level window takes precedence over a file-level or global window for the dates associated with that field.

For details on Master Files, see your documentation on describing data.

Syntax

How to Define a File-Level Window in a Master File

To define a window that applies to all legacy date fields in a file, add the FDEFCENT and FYRTHRESH attributes to the Master File on the file declaration.

The syntax for the first attribute is

\[
\text{FDEFCENT|FDFC} = \{ cc | 19 \}
\]

where:

- \( cc \)
  
  Is the century for the start date of the window. If you do not supply a value, \( cc \) defaults to 19, for the twentieth century.
Working With Cross-Century Dates

The syntax for the second attribute is

\{FYRTHRESH|FYRT\} = \{[-]yy|0\}

where:

\(yy\)

Is the year threshold for the window. If you do not supply a value, \(yy\) defaults to zero (0).

If \(yy\) is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If \(yy\) is a negative number (-\(yy\)), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

Example

Defining a File-Level Window in a Master File

Tip: Use the abbreviated forms of FDEFCENT/FYRTHRESH or DEFCENT/YRTHRESH to reduce keystrokes. The examples in this topic use the abbreviated forms where available (for instance, FDFC instead of FDEFCENT). Maintain supports only the abbreviated forms in certain command syntax (for example, on a COMPUTE or DECLARE command). For details, see your database maintenance documentation.

In the following example, the FDEFCENT and FYRTHRESH attributes define a window from 1982 to 2081. The window is applied to all legacy date fields in the file, including HIRE_DATE, DAT_INC, and others, if they are converted to a date format.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
SEGNAME=EMPINFO, SEGTYPE=S1
FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, $
FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, $
```
The DEFINE command in the following request creates two virtual fields named NEW_HIRE_DATE, which is derived from the existing field HIRE_DATE, and NEW_DAT_INC, which is derived from DAT_INC. The format of HIRE_DATE and DAT_INC is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE and NEW_DAT_INC are date formats with 4-digit years (YYMD). For details on DEFINE, see your documentation on creating reports.

```
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END
```

```
TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END
```

The window created in the Master File applies to both legacy date fields. In the report, the year 82 (which is equal to the threshold), for both HIRE_DATE and DAT_INC, defaults to the century value 19 and is returned as 1982 in the NEW_HIRE_DATE and NEW_DAT_INC columns. The year 81, for both HIRE_DATE and DAT_INC, is less than the threshold 82 and assumes the century value 20 (FDEFCENT + 1).

The partial report is:

```
PAGE 1
HIRE_DATE   NEW_HIRE_DATE   DAT_INC   NEW_DAT_INC
---------   -------------   -------   -----------
80/06/02   2080/06/02     82/01/01   1982/01/01
80/06/02   2080/06/02     81/01/01   2081/01/01
81/07/01   2081/07/01     82/01/01   1982/01/01
82/05/01   1982/05/01     82/06/01   1982/06/01
82/05/01   1982/05/01     82/05/01   1982/05/01
```

Developing Applications
Working With Cross-Century Dates

**Syntax**

How to Define a Field-Level Window in a Master File

To define a window that applies to a specific legacy date field, add the DEFCENT and YRTHRESH attributes to the Master File on the field declaration.

The syntax for the first attribute is

\[
\text{(DEFCENT|DFC) = \{cc\|19}
\]

where:

- \( cc \)
  - Is the century for the start date of the window. If you do not supply a value, \( cc \) defaults to 19, for the twentieth century.

The syntax for the second attribute is

\[
\text{(YRTHRESH|YRT) = \{[-]\ yy\|0}
\]

where:

- \( yy \)
  - Is the year threshold for the window. If you do not supply a value, \( yy \) defaults to zero (0).

  - If \( yy \) is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of \( \text{DEFCENT} + 1 \).

  - If \( yy \) is a negative number (-\( yy \)), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

**Example**

Defining a Field-Level Window in a Master File

In this example, the application requires a different window for two legacy date fields in the same file.

The DEFCENT and YRTHRESH attributes in the Master File define a window for HIRE_DATE from 1982 to 2081, and a window for DAT_INC from 1983 to 2082.

The Master File is:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $  
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $  
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $  
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, DFC=19, YRT=82, $  
  .  
  .  
  .  
  FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, DFC=19, YRT=83, $  
  .  
  .  
```

1-16 Information Builders
Defining a File-Level or Field-Level Window in a Master File

The request is the same one used in the previous example (defining a file-level window in a Master File):

DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE DAT_INC NEW_DAT_INC
END

However, the report illustrates the use of two different windows for the two legacy date fields. For example, the year 82 for HIRE_DATE defaults to the century value 19, since 82 is equal to the threshold for the window for this field. The date returned for NEW_HIRE_DATE is 1982.

The year 82 for DAT_INC assumes the century value 20 (DEFCENT + 1), since 82 is less than the threshold for the window for this field (83). The date returned for NEW_DAT_INC is 2082.

The partial report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRE_DATE</td>
<td>NEW_HIRE_DATE</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/06/02</td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/06/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/01</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>
**Example**

**Defining a Field-Level Window in a Master File Used With MODIFY**

This example illustrates the use of field-level DEFCENT and YRTHRESH attributes to define a window used with MODIFY. To run this example yourself, you need to create a Master File named DATE and a procedure named DATELOAD.

The Master File describes a segment with 12 date fields of different formats. The first field is a date format field. The DEFCENT and YRTHRESH attributes included on this field create a window from 1990 to 2089. The window is required because the input data for the first date field does not contain century digits, and the default value 19 cannot be assumed.

The Master File looks like this:

```plaintext
FILENAME=DATE, SUFFIX=FOC
SEGNAME=ONE, SEGTYPE=S1
FIELDNAME=D1_YYMD, ALIAS=D1, FORMAT=YYMD, DFC=19, YRT=90, $
FIELDNAME=D2_I6YMD, ALIAS=D2, FORMAT=I6YMD, $
FIELDNAME=D3_I8YYMD, ALIAS=D3, FORMAT=I8, $
FIELDNAME=D4_A6YMD, ALIAS=D4, FORMAT=A6YMD, $
FIELDNAME=D5_A8YYMD, ALIAS=D5, FORMAT=A8YYMD, $
FIELDNAME=D6_I4YM, ALIAS=D6, FORMAT=I4YM, $
FIELDNAME=D7_YQ, ALIAS=D7, FORMAT=YQ, $
FIELDNAME=D8_YM, ALIAS=D8, FORMAT=YM, $
FIELDNAME=D9_JUL, ALIAS=D9, FORMAT=JUL, $
FIELDNAME=D10_Y, ALIAS=D10, FORMAT=Y, $
FIELDNAME=D11_YY, ALIAS=D11, FORMAT=YY, $
FIELDNAME=D12_MDYY, ALIAS=D12, FORMAT=MDYY, $
```

The procedure (DATELOAD) creates a FOCUS data source named DATE and loads two records into it. The first field of the first record contains the 2-digit year 92. The first field of the second record contains the 2-digit year 88. For details on commands such as CREATE and MODIFY, and others used in this file, see your database maintenance documentation.

The procedure looks like this:

```plaintext
CREATE FILE DATE
MODIFY FILE DATE
MATCH D1
  ON NOMATCH INCLUDE
  ON MATCH REJECT
DATA
  92022900022920000229000229200002290002000100006000200002292000
  88022900022920000229000229200002290002000100006000200002292000
END
```

The following request accesses all the fields in the new data source:

```plaintext
TABLE FILE DATE
PRINT *
END
```
Defining a File-Level or Field-Level Window in a Master File

In the report, the year 92 for D1_YYMD defaults to the century value 19, since 92 is greater than the threshold for the window for this field (90). It is returned as 1992 in the D1_YYMD column. The year 88 assumes the century value 20 (DEFCENT + 1), because 88 is less than the threshold. It is returned as 2088 in the D1_YYMD column.

The partial report is:

```
<table>
<thead>
<tr>
<th>PAGE</th>
<th>D1_YYMD</th>
<th>D2_I6YMD</th>
<th>D3_I8YMD</th>
<th>D4_A6YMD</th>
<th>D5_A8YYMD</th>
<th>D6_I4YM</th>
<th>D7_YQ</th>
<th>D8_YM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1992/02/29</td>
<td>00/02/29</td>
<td>20000229</td>
<td>00/02/29</td>
<td>2000/02/29</td>
<td>00/02</td>
<td>Q1</td>
<td>00/02</td>
</tr>
<tr>
<td>2</td>
<td>2088/02/29</td>
<td>00/02/29</td>
<td>20000229</td>
<td>00/02/29</td>
<td>2000/02/29</td>
<td>00/02</td>
<td>Q1</td>
<td>00/02</td>
</tr>
</tbody>
</table>
```

Example: Defining Both File-Level and Field-Level Windows

The following Master File defines windows at both the file and field level:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=83
SEGNAME=EMPINFO, SEGTYPE=S1
  FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
  FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
  FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
  FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, DFC=19, YRT=82, $
  FIELDNAME=EFFECT_DATE, ALIAS=EDATE, FORMAT=I6YMD, $
  FIELDNAME=DAT_INC, ALIAS=DI, FORMAT=I6YMD, $

The request is:
DEFINE FILE EMPLOYEE
NEW_HIRE_DATE/YYMD = HIRE_DATE;
NEW_EFFECT_DATE/YYMD = EFFECT_DATE;
NEW_DAT_INC/YYMD = DAT_INC;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE EFFECT_DATE NEW_EFFECT_DATE DAT_INC NEW_DAT_INC
END
```

When the field HIRE_DATE is accessed, the time span 1982 to 2081 is applied. For all other legacy date fields in the file, such as EFFECT_DATE and DAT_INC, the time span specified at the file level is applied, that is, 1983 to 2082.
Working With Cross-Century Dates

For example, the year 82 for HIRE_DATE is returned as 1982 in the NEW_HIRE_DATE column, since 82 is equal to the threshold of the window for that particular field. The year 82 for EFFECT_DATE and DAT_INC is returned as 2082 in the columns NEW_EFFECT_DATE and NEW_DAT_INC, since 82 is less than the threshold of the file-level window (83).

The partial report is:

```
<table>
<thead>
<tr>
<th>HIRE_DATE</th>
<th>NEW_HIRE_DATE</th>
<th>EFFECT_DATE</th>
<th>NEW_EFFECT_DATE</th>
<th>DAT_INC</th>
<th>NEW_DAT_INC</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/06/02</td>
<td>2080/06/02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/06/02</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
<td>82/11/01</td>
<td>2082/11/01</td>
<td>82/05/01</td>
<td>2082/05/01</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
<td>82/11/01</td>
<td>2082/11/01</td>
<td>82/05/01</td>
<td>2082/05/01</td>
</tr>
</tbody>
</table>
```

Missing date values for NEW_EFFECT_DATE appear as blanks by default. To retrieve the base date value for NEW_EFFECT_DATE instead of blanks, issue the command

```
SET DATEDISPLAY = ON
```

before running the request. The base date value is returned as 1900/12/31. See the last example in Defining a Global Window With SET on page 1-6 for sample results.

Defining a Window for a Virtual Field

The DEFCENT and YRTHRESH parameters on a DEFINE command create a window for a virtual field. The window is used to interpret date values for the virtual field when the century is not supplied. You can issue a DEFINE command in either a request or a Master File.

The DEFCENT and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the DEFINE syntax (that is, from the left side of the equal sign). If the expression in the DEFINE contains a subroutine call, the subroutine uses the DEFCENT and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCENT and YRTHRESH values for the input field.
Defining a Window for a Virtual Field

Syntax

How to Define a Window for a Virtual Field in a Request

Use standard DEFINE syntax for a request, as described in your documentation on creating reports. Partial DEFINE syntax is shown here.

On the line that specifies the name of the virtual field, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
DEFINE FILE filename
    fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] =
    expression;
.
.
END
```

where:

- **filename**
  - Is the name of the file for which you are creating the virtual field.

- **fieldname**
  - Is the name of the virtual field.

- **format**
  - Is a date format such as DMY or YYMD.

- **DEFCENT**
  - Is the parameter for the default century.

- **cc**
  - Is the century for the start date of the window. If you do not supply a value, cc defaults to 19, for the twentieth century.

- **YRTHRESH**
  - Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.

- **yy**
  - Is the year threshold for the window. If you do not supply a value, yy defaults to zero (0).

  If yy is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

  If yy is a negative number (-yy), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.
Working With Cross-Century Dates

expression
Is a valid arithmetic or logical expression, function, or subroutine that determines the value of the virtual field.

END
Is required to terminate the DEFINE command.

Example
Defining a Window for a Virtual Field in a Request

In the following request, the DEFINE command creates two virtual fields, GLOBAL_HIRE_DATE and WINDOWED_HIRE_DATE. Both virtual fields are derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. The virtual fields are date formats with a 4-digit year (YYMD).

The second virtual field, WINDOWED_HIRE_DATE, has the additional parameters DEFCENT and YRTHRESH, which define a window from 1982 to 2081. Notice that both DEFCENT and YRTHRESH are coded, as required.

The request is:

DEFINE FILE EMPLOYEE
GLOBAL_HIRE_DATE/YYMD = HIRE_DATE;
WINDOWED_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE GLOBAL_HIRE_DATE WINDOWED_HIRE_DATE
END

Assuming that there are no FDEFCENT and FYRTHRESH file-level settings in the Master File for EMPLOYEE, the global default settings (DEFCENT = 19, YRTHRESH = 0) are used to interpret 2-digit years for HIRE_DATE when deriving the value of GLOBAL_HIRE_DATE. For example, the value of all years for HIRE_DATE (80, 81, and 82) is greater than 0; consequently they default to 19 for the century and are returned as 1980, 1981, and 1982 in the GLOBAL_HIRE_DATE column.

For WINDOWED_HIRE_DATE, the window created specifically for that field (1982 to 2081) is used. The 2-digit years 80 and 81 for HIRE_DATE are less than the threshold for the window (82); consequently, they are returned as 2080 and 2081 in the WINDOWED_HIRE_DATE column.
The report is:

<table>
<thead>
<tr>
<th>HIRE_DATE</th>
<th>GLOBAL_HIRE_DATE</th>
<th>WINDOWED_HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>80/06/02</td>
<td>1980/06/02</td>
<td>2080/06/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>1981/07/01</td>
<td>2081/07/01</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
<td>2082/05/01</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/04</td>
<td>2082/01/04</td>
</tr>
<tr>
<td>82/08/01</td>
<td>1982/08/01</td>
<td>2082/08/01</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/04</td>
<td>2082/01/04</td>
</tr>
<tr>
<td>82/07/01</td>
<td>1982/07/01</td>
<td>2082/07/01</td>
</tr>
<tr>
<td>81/07/01</td>
<td>1981/07/01</td>
<td>2081/07/01</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/01</td>
<td>2082/04/01</td>
</tr>
<tr>
<td>82/02/02</td>
<td>1982/02/02</td>
<td>2082/02/02</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/01</td>
<td>2082/04/01</td>
</tr>
<tr>
<td>81/11/02</td>
<td>1981/11/02</td>
<td>2081/11/02</td>
</tr>
</tbody>
</table>

**Example**

**Defining a Window for Subroutine Input in a DEFINE Command**

The following sample request illustrates a call to the subroutine AYMD in a DEFINE command. AYMD adds 60 days to the input field, HIRE_DATE; the output field, SIXTY_DAYS, contains the result. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. SIXTY_DAYS is formatted as I8YYMD, which is a legacy date with a 4-digit year.

For details on AYMD, see your documentation on creating reports.

```
DEFINE FILE EMPLOYEE
SIXTY_DAYS/I8YYMD = AYMD(HIRE_DATE, 60, 'I8YYMD');
END

TABLE FILE EMPLOYEE
PRINT HIRE_DATE SIXTY_DAYS
END
```

The subroutine uses the DEFCENT and YRTHRESH values for the input field HIRE_DATE. In this example, they are set on the field level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9, $
FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15, $
FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10, $
FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD, DFC=19, YRT=82, $
```

The subroutine inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.
## Working With Cross-Century Dates

The report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRE_DATE</td>
<td>SIXTY_DAYS</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/08/01</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/08/30</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/06/30</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/03/05</td>
</tr>
<tr>
<td>82/08/01</td>
<td>1982/09/30</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/03/05</td>
</tr>
<tr>
<td>82/07/01</td>
<td>1982/08/30</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/08/30</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/05/31</td>
</tr>
<tr>
<td>82/02/02</td>
<td>1982/04/03</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/05/31</td>
</tr>
<tr>
<td>81/11/02</td>
<td>2082/01/01</td>
</tr>
</tbody>
</table>

### Syntax

**How to Define a Window for a Virtual Field in a Master File**

Use standard DEFINE syntax for a Master File, as discussed in your documentation on describing data. Partial DEFINE syntax is shown here.

The parameters DEFCENT and YRTHRESH must immediately follow the field format information.

```plaintext
DEFINE fieldname/[format] {DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {-|yy|0} = expression;$
```

where:

- `fieldname` is the name of the virtual field.
- `format` is a date format such as DMY or YYMD.
- `DEFCENT` is the parameter for the default century.
- `cc` is the century for the start date of the window. If you do not supply a value, `cc` defaults to 19, for the twentieth century.
- `YRTHRESH` is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.
Defining a Window for a Virtual Field

yy

Is the year threshold for the window. If you do not supply a value, yy defaults to zero (0).
If yy is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.
If yy is a negative number (-yy), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.

expression

Is a valid arithmetic or logical expression, function, or subroutine that determines the value of the virtual field.

Example

Defining a Window for a Virtual Field in a Master File

In the following example, the DEFINE command in a Master File creates a virtual field named NEW_HIRE_DATE. It is derived from the existing field HIRE_DATE. The format of HIRE_DATE is I6YMD, which is a legacy date with a 2-digit year. NEW_HIRE_DATE is a date format with a 4-digit year (YYMD).

The parameters DEFCENT and YRTHRESH on the DEFINE command create a window from 1982 to 2081, which is used to interpret all 2-digit years for the virtual field. Notice that both DEFCENT and YRTHRESH are coded, as required.

The field-level window takes precedence over any global settings in effect. There is no file-level setting in the Master File.

The Master File is:

FILENAME=EMPLOYEE, SUFFIX=FOC
SEGNAME=EMPINFO, SEGTYPE=S1
FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9,
FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15,
FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10,
FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD,
.
.
DEFINE NEW_HIRE_DATE/YYMD DFC 19 YRT 82 = HIRE_DATE;$

The following request generates the values in the sample report:

TABLE FILE EMPLOYEE
PRINT HIRE_DATE NEW_HIRE_DATE
END

Since the 2-digit years 80 and 81 are less than the threshold 82, their century assumes the value of DEFCENT + 1 (20), and they are returned as 2080 and 2081 in the NEW_HIRE_DATE column. The 2-digit year 82 is equal to the threshold and therefore defaults to the value of DEFCENT (19). It is returned as 1982.
Working With Cross-Century Dates

The report is:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>NEW_HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIRE_DATE</td>
<td>--------------</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>80/06/02</td>
<td>2080/06/02</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/01</td>
</tr>
<tr>
<td>82/05/01</td>
<td>1982/05/01</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/04</td>
</tr>
<tr>
<td>82/08/01</td>
<td>1982/08/01</td>
</tr>
<tr>
<td>82/01/04</td>
<td>1982/01/04</td>
</tr>
<tr>
<td>82/07/01</td>
<td>1982/07/01</td>
</tr>
<tr>
<td>81/07/01</td>
<td>2081/07/01</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/01</td>
</tr>
<tr>
<td>82/02/02</td>
<td>1982/02/02</td>
</tr>
<tr>
<td>82/04/01</td>
<td>1982/04/01</td>
</tr>
<tr>
<td>81/11/02</td>
<td>2081/11/02</td>
</tr>
</tbody>
</table>

Defining a Window for a Temporary Field

Use the DEFCENT and YRTHRESH parameters on a COMPUTE command in a report request to create a window for a temporary field that is calculated from the result of a PRINT, LIST, SUM, or COUNT command. The window is used to interpret a date value for that field when the century is not supplied.

You can also use the parameters on a COMPUTE command in a MODIFY or Maintain procedure, or on a DECLARE command in Maintain. For details on the use of the parameters in Maintain, see your database maintenance documentation.

The DEFCENT and YRTHRESH parameters must immediately follow the field format specification; their values are always taken from the left side of the COMPUTE syntax (that is, from the left side of the equal sign). If the expression in the COMPUTE contains a subroutine call, the subroutine uses the DEFCENT and YRTHRESH values for the input field. The standard order of precedence (field level/file level/global level) applies to the DEFCENT and YRTHRESH values for the input field.

Some development environments refer to a virtual field created by DEFINE as a temporary field. However, this topic refers to a field created by DEFINE only as a virtual field, and to a field created by COMPUTE as a temporary field. For details on virtual fields created by DEFINE, see Defining a Window for a Virtual Field on page 1-20.
Defining a Window for a Temporary Field

Syntax

How to Define a Window for a Temporary Field in a Report

Use standard COMPUTE syntax, as described in your documentation on creating reports. Partial COMPUTE syntax is shown here.

On the line that specifies the name of the temporary field, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
TABLE FILE filename
command
[AND] COMPUTE
   fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] = expression;
 .
 .
END
```

where:

- `filename` is the name of the file for which you are creating the temporary field.
- `command` is a command such as PRINT, LIST, SUM, or COUNT.
- `fieldname` is the name of the temporary field.
- `format` is a date format such as DMY or YYMD.
- `DEFCENT` is the parameter for the default century.
- `cc` is the century for the start date of the window. If you do not supply a value, `cc` defaults to 19, for the twentieth century.
- `YRTHRESH` is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.
- `yy` is the year threshold for the window. If you do not supply a value, `yy` defaults to zero (0).

If `yy` is a positive number, two-digit years greater than or equal to the threshold default to the value of DEFCENT for the century. Two-digit years less than the threshold assume the value of DEFCENT + 1.

If `yy` is a negative number (-yy), the start date of the window is derived by subtracting that number from the current year, and DEFCENT is automatically calculated. The start date is automatically incremented by one at the beginning of each successive year.
Working With Cross-Century Dates

expression
Is a valid arithmetic or logical expression, function, or subroutine that determines the value of the temporary field.

END
Is required to terminate the request.

**Syntax**

**How to Define a Window for a Temporary Field in a MODIFY Request**

Use standard MODIFY and COMPUTE syntax, as described in your database maintenance documentation; partial syntax is shown here.

On the line that specifies the name of the temporary field, include the DEFCENT and YRTHRESH parameters and values. The parameters must immediately follow the field format information.

```
MODIFY FILE filename
.
.
.
COMPUTE
  \[fieldname[/format] [{DEFCENT|DFC} {cc|19} {YRTHRESH|YRT} {[-]yy|0}] = expression;
  .
  .
  .
[END]
```

where:

- `filename`
  Is the name of the file you are modifying.

- `fieldname`
  Is the name of the field being set to the value of `expression`.

- `format`
  Is a date format such as MDY or YYMD.

- `DEFCENT`
  Is the parameter for the default century.

- `cc`
  Is the century for the start date of the window. If you do not supply a value, `cc` defaults to 19, for the twentieth century.

- `YRTHRESH`
  Is the parameter for the year threshold. You must code values for both DEFCENT and YRTHRESH unless YRTHRESH is negative. In that case, only code a value for YRTHRESH.
Defining a Window for a Temporary Field

.yy
Is the year threshold for the window. If you do not supply a value, yy defaults to zero (0).
If yy is a positive number, two-digit years greater than or equal to the threshold default to
the value of DEFCENT for the century. Two-digit years less than the threshold assume the
value of DEFCENT + 1.
If yy is a negative number (-yy), the start date of the window is derived by subtracting that
number from the current year, and DEFCENT is automatically calculated. The start date is
automatically incremented by one at the beginning of each successive year.

.expression
Is a valid arithmetic or logical expression, function, or subroutine that determines the value
of fieldname.

.END
Terminates the request. Do not add this command if the request contains PROMPT
statements.

Example Defining a Window for a Temporary Field

In the following request, the parameters DEFCENT and YRTHRESH on the COMPUTE
command define a window from 1999 to 2098. Notice that both DEFCENT and YRTHRESH
are coded, as required. The window is applied to the field created by the COMPUTE command,
LATEST_DAT_INC.

DAT_INC is formatted as I6YMD, which is a legacy date with a 2-digit year.
LATEST_DAT_INC is a date format with a 4-digit year (YYMD). The prefix MAX retrieves
the highest value of DAT_INC.

The request is:

TABLE FILE EMPLOYEE
SUM SALARY AND COMPUTE
  LATEST_DAT_INC/YYMD DFC 19 YRT 99 = MAX.DAT_INC;
END

The highest value of DAT_INC is 82/08/01. Since the year 82 is less than the threshold 99, it
assumes the value 20 for the century (DEFCENT + 1).

The report is:

PAGE 1

<table>
<thead>
<tr>
<th>SALARY</th>
<th>LATEST_DAT_INC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$332,929.00</td>
<td>2082/08/01</td>
</tr>
</tbody>
</table>
**Example**

**Defining a Window for Subroutine Input in a COMPUTE Command**

The following sample request illustrates a call to the subroutine JULDAT in a COMPUTE command. JULDAT converts dates from Gregorian format (year/month/day) to Julian format (year/day). For century display, dates in Julian format are 7-digit numbers. The first 4 digits are the century. The last three digits represent the number of days, counting from January 1.

For details on JULDAT, see your documentation on creating reports.

In the request, the input field is HIRE_DATE. The subroutine converts it to Julian format and returns it as JULIAN_DATE. HIRE_DATE is formatted as I6YMD, which is a legacy date with a 2-digit year. JULIAN_DATE is formatted as I7, which is a legacy date with a 4-digit year.

```
TABLE FILE EMPLOYEE
PRINT DEPARTMENT HIRE_DATE
AND COMPUTE
   JULIAN_DATE/I7 = JULDAT(HIRE_DATE, JULIAN_DATE);
END
```

The subroutine uses the FDEFCENT and FYRTHRESH values for the input field HIRE_DATE. In this example, they are set on the file level in the Master File:

```
FILENAME=EMPLOYEE, SUFFIX=FOC, FDFC=19, FYRT=82
SEGNAME=EMPINFO, SEGTYPE=S1
FIELDNAME=EMP_ID, ALIAS=EID, FORMAT=A9,
FIELDNAME=LAST_NAME, ALIAS=LN, FORMAT=A15,
FIELDNAME=FIRST_NAME, ALIAS=FN, FORMAT=A10,
FIELDNAME=HIRE_DATE, ALIAS=HDT, FORMAT=I6YMD,
```

The subroutine inputs a 2-digit year, which is windowed. It then outputs a 4-digit year that includes the century digits.

The input values 80 and 81 are less than the threshold 82, so they assume the value 20 for the century. The input value 82 is equal to the threshold, so it defaults to 19 for the century.
The report follows. By default, the second occurrence of the last name SMITH displays as blanks.

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DEPARTMENT</th>
<th>HIRE_DATE</th>
<th>JULIAN_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>PRODUCTION</td>
<td>82/08/01</td>
<td>1982213</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>MIS</td>
<td>82/04/01</td>
<td>1982091</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>MIS</td>
<td>81/11/02</td>
<td>2081306</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>MIS</td>
<td>82/04/01</td>
<td>1982091</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOAN</td>
<td>PRODUCTION</td>
<td>82/01/04</td>
<td>1982004</td>
</tr>
<tr>
<td>JONES</td>
<td>DIANE</td>
<td>MIS</td>
<td>82/05/01</td>
<td>1982121</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>MIS</td>
<td>81/07/01</td>
<td>2081182</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>PRODUCTION</td>
<td>82/02/02</td>
<td>1982033</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>PRODUCTION</td>
<td>82/07/01</td>
<td>1982182</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>MIS</td>
<td>81/07/01</td>
<td>2081182</td>
</tr>
<tr>
<td>ROMANS</td>
<td>RICHARD</td>
<td>PRODUCTION</td>
<td>82/01/04</td>
<td>1982004</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>PRODUCTION</td>
<td>80/06/02</td>
<td>2080154</td>
</tr>
</tbody>
</table>

Additional Support for Cross-Century Dates

The following features apply to the use of dates in your applications.

Default Date Display Format

The default date display format is MM/DD/CCYY, where MM is the month; DD is the day of the month; CC is the first two digits of a 4-digit year, indicating the century; and YY is the last two digits of a 4-digit year.

For example:

02/11/1999

For a table that fully describes the display of a date based on the specified format and user input, see your documentation on describing data.
**Date Display Options**

The following date display options are available:

- You can display a row of data, even though it contains an invalid date field, using the command SET ALLOWC VTERR. The invalid date field is returned as the base date or as blanks, depending on other settings. For details, see your documentation on the SET command. This feature applies to non-FOCUS data sources when converting from the way data is stored (ACTUAL attribute) to the way it is formatted (FORMAT or USAGE attribute).

- If a date format field contains the value zero (0), you can display its base date, using the command SET DATEDISPLAY = ON. By default, the value zero in a date format field such as YYMD is returned as a blank. For details, see your documentation on the SET command.

- You can display the current date with a 4-digit year using the Dialogue Manager system variables &YYMD, &MDYY, and &DMYY. The system variable &DATEfmt displays the current date as specified by the value of fmt, which is a combination of allowable date options, including a 4-digit year (for example, &DATEYYMD). For details, see your documentation on Dialogue Manager.

**System Date Masking**

You can temporarily alter the system date for application testing and debugging, using the command SET TESTDATE. With this feature, you can simulate clock settings beyond the year 1999 to determine the way your program will behave. For details, see your documentation on the SET command.

**Date Functions and Subroutines**

The date functions and subroutines supplied with your software work across centuries. Many of them facilitate date manipulation. For details on date functions and subroutines, see your documentation on creating reports.
Additional Support for Cross-Century Dates

Date Conversion
You can convert a legacy date to a date format in a FOCUS data source using the option DATE NEW on the REBUILD command. For details, see your documentation on database maintenance.

Century and Threshold Information
The ALL option, in conjunction with the HOLD option, on the CHECK FILE command includes file-level and field-level default century and year thresholds as specified in a Master File. For details, see your documentation on describing data.

Date Time Stamp
The year in the time stamp for a FOCUS data source is physically written to page one of the file in the format CCYY.
Topics:
- Overview of Dialogue Manager Capabilities
- Creating and Storing Procedures
- Executing Procedures
- Including Comments in a Procedure
- Overview of Dialogue Manager Commands
- Sending a Message to the User: -TYPE
- Controlling Execution: -RUN, -EXIT, and -QUIT
- Branching
- Looping
- Using Expressions: -SET
- Additional Facilities
- Using Variables in Procedures
- Supplying Values for Variables at Run Time
- Dialogue Manager Quick Reference

Dialogue Manager enables you to execute stored procedures. In the FOCUS community, stored procedures are referred to as FOCEXECs. In this chapter, they are referred to simply as procedures.

Dialogue Manager helps you build and manage the execution of procedures, giving you flexibility in application design. You can use Dialogue Manager control statements to determine the sequence in which FOCUS commands (such as TABLE) execute. Dialogue Manager also enables you to use variables in your procedures and supply values for those variables at run time. You can create a dialogue between the user and the terminal through various prompting methods, including full-screen forms, menus and windows that you design yourself, and system queries, as well as supplying values directly in the procedure.
Managing Applications With Dialogue Manager

Overview of Dialogue Manager Capabilities

Using Dialogue Manager control statements and variables, your application can respond to user input and environment conditions at run time. It is important to understand how Dialogue Manager processes an application’s commands and variables.

**Example** Processing a Procedure

The following example traces the execution process of a procedure. The numbers at the left refer to explanatory notes that follow the example.

1. -TOP
2. -PROMPT &WHICH_CITY. ENTER NAME OF CITY OR DONE.
3. -IF &WHICH_CITY EQ 'DONE' GOTO QUIT;
4. TABLE FILE SALES
   SUM UNIT_SOLD
   BY PROD_CODE
   IF CITY IS &WHICH_CITY
   END
5. -RUN
6. -GOTO TOP
7. -QUIT

Assume that this procedure is stored in a file named SLRPT. To execute it, the user types either of the following:

EXEC SLRPT

or

EX SLRPT

The following describes the individual steps of the procedure:

1. -TOP

   This is a label, which serves as a target to which -IF … GOTO or -GOTO commands transfer processing control. Labels themselves call for no special processing, so in this case control passes to the next command.

2. -PROMPT &WHICH_CITY. ENTER NAME OF CITY OR DONE.

   The prompt “ENTER NAME OF CITY OR DONE” appears on the terminal. Assume the user types “STAMFORD” and the variable value is stored for later use. Processing continues with the next line.

3. -IF &WHICH_CITY EQ 'DONE' GOTO QUIT;

   Had DONE been entered, control would pass to -QUIT at the bottom of the procedure. This would end processing, cause an immediate exit from this procedure, and return control to the FOCUS prompt. Since STAMFORD was entered, processing continues with the next line.
4. **TABLE FILE SALES**
   
   -
   -
   -

   Since there is no leading hyphen, this is interpreted as a FOCUS command. Only Dialogue Manager commands execute immediately, so the next five lines are placed in the stack where FOCUS commands are kept until executed; this is referred to as FOCSTACK. Note that the value STAMFORD, entered in response to the prompt, is inserted into the FOCUS command line as the value for &WHICHCITY.

   At this point the FOCSTACK looks like this:

   **TABLE FILE SALES**
   **SUM UNIT_SOLD**
   **BY PROD_CODE**
   **IF CITY IS STAMFORD**
   **END**

   Control passes to the next Dialogue Manager command.

5. **-RUN**

   This command sends the stack to FOCUS, which executes the stored request and returns control to the next Dialogue Manager command.

6. **-GOTO TOP**

   Control is now routed back to -TOP, thus establishing a loop. Execution continues from -TOP with the -PROMPT command.
Managing Applications With Dialogue Manager

7. -QUIT

This command is reached when the user types DONE in response to the prompt. The procedure is exited and the FOCUS prompt appears.

---

**Figure 2-1. Schematic Diagram of Procedure Processing**

1. Processing begins from the command processor when a procedure is invoked for execution at the FOCUS prompt (for example, EX SLRPT).

2. The FOCEXEC Processor reads each line of the procedure. Any variables on the line are assigned their current values.

3. If a variable is missing a value, FOCUS issues a prompt. The user then supplies the missing value.

4. When a command line that contains no Dialogue Manager commands is fully expanded with any variables resolved (through either a -SET command or prompting), it is placed onto the command execution stack (FOCSTACK).

5. Dialogue Manager execution commands (for example, -RUN) and statistical variables flush the FOCSTACK and route all currently stacked commands to the FOCUS Command Processor.
6. In the previous example the FOCUS Command Processor routes execution to the TABLE module and executes the TABLE request that was stacked.

By the time your FOCSTACK is ready for execution, this has happened:

- All variables have received values and these values have been integrated into the command lines containing variables.
- Dialogue Manager commands have been used to place FOCUS commands into proper sequential order for execution.
- At this point the FOCUS Command Processor no longer sees any Dialogue Manager commands. It only sees FOCUS command lines in the stack.

Note: Any FOCUS command can be placed in a procedure. This includes the EXEC command itself. When an EXEC command is processed in a procedure, the commands from the new procedure are first stacked and then executed. Multiple levels of nesting are permitted when you use the EXEC command, while only four levels of nesting are permitted when you use -INCLUDE.

Overview of Dialogue Manager Variables

You can write procedures that contain variables whose values are unknown until run time; this technique allows a user to customize the procedure by supplying different values each time it executes. Variables fall into two categories:

- Local and global variables, whose values must be supplied at run time. Local variables retain their values only for one procedure. Global variables retain their values across procedures unless you explicitly clear them. They lose their values when you exit from FOCUS. You create a local variable by choosing a name that starts with a single ampersand (&); you create a global variable by choosing a name that starts with a double ampersand (&&).

- System and statistical variables, whose values are automatically supplied by the system when a procedure references them. System and statistical variables have names that begin with a single ampersand (&). For example, the variable &LINES indicates how many lines of output were produced, and the variable &DATE indicates the current date.

For complete information about variables, see *Using Variables in Procedures* on page 2-46.
Creating and Storing Procedures

You can create procedures with your system editor or with the FOCUS integrated text editor, TED. TED has two features that make it particularly useful for creating and editing procedures:

- If you type TED without specifying a procedure name, the last executed procedure is automatically selected. This is convenient when developing and testing new procedures.
- You can test the execution of the procedure by typing RUN on the TED command line. This automatically saves the procedure and executes it. If there is an error in your procedure, type TED. This brings you back into the editor and places you directly on the line in which the error was detected.

These options complement the FILE and SAVE options that are common to other editors.

Follow these general rules when you are creating procedures:

- Dialogue Manager commands must begin in the first position of the line.
- At least one space must be inserted between the Dialogue Manager command and other text.
- If a Dialogue Manager command exceeds one line, the following line must begin with a hyphen (-) in the first position. The continuation line can begin immediately after the hyphen, or you may insert a space between the hyphen and the rest of the line.
- Procedures must have the record format RECFM=F and the logical record length (LRECL) 80.

Executing Procedures

Procedures are generally initiated from the FOCUS prompt (>). Type the command EXEC, or its abbreviation EX, followed by the name of the procedure.

Example

Executing a Procedure

Either of the following commands

EXEC SLRPT
or

EX SLRPT

will summon the procedure named SLRPT for execution.
Controlling Access to Data

You can set a password in a procedure and tie it to different portions of a procedure.

Syntax

How to Set a Password in a Procedure

The syntax is

```
-PASS password
```

where:

password

Is a password or a variable containing a password.

Since -PASS is a Dialogue Manager command, it executes immediately and is not sent to the FOCSTACK. This means that the user need not issue the password with the SET command.

Including Comments in a Procedure

It is good practice to include comments in procedures for the benefit of others who may read or refine them at a later date. Comments are particularly recommended as procedure headers to give version, date, and other relevant information. It is easier to track and maintain large software applications when they are carefully commented. Comments are ignored during actual execution.

To add comment lines to a Dialogue Manager procedure, precede them with a hyphen and an asterisk (*). Any text whatsoever may immediately follow the -*. You can place comment lines anywhere in a procedure.

Comments do not appear on the terminal nor do they trigger any processing. They are visible only when viewing the contents of the procedure through the editor and are strictly for the benefit of the developer. However, you can view comments on the terminal by using the option ECHO = ALL.

Example

Including Comments in a Procedure

The following example contains two comment lines:

```
-* Version 1 6/30/85 SLRPT FOCEXEC
-* Component of Retail Sales Reporting Module
TABLE FILE SALES
HEADING CENTER
"MONTHLY SALES FOR STAMFORD"
```

Developing Applications 2-7
Overview of Dialogue Manager Commands

Dialogue Manager provides commands for accomplishing the following tasks:

- Sending messages to the user.
- Displaying values.
- Controlling the values of variables, including reading variables from and writing values to an external file.
- Testing conditions and branching.
- Controlling the execution of stacked commands.
- Calling another procedure.
- Issuing operating system commands specific to your environment.

Reference Summary of Dialogue Manager Commands

The following pages describe all Dialogue Manager commands. They are listed in alphabetical order. The categories used to describe them in the quick reference at the end of this chapter are briefly outlined below:

<table>
<thead>
<tr>
<th>Command:</th>
<th>Lists the name of the command.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>Shows exactly how the command components must appear in a procedure.</td>
</tr>
<tr>
<td>Function:</td>
<td>Outlines the meaning and purpose of the command.</td>
</tr>
<tr>
<td>Similar Command:</td>
<td>Describes the relationship between the Dialogue Manager command and other FOCUS commands (for example, -TYPE and TYPE).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-*</td>
<td>Is a comment line; it has no action.</td>
</tr>
<tr>
<td>-CLOSE ddname</td>
<td>Closes the specified -READ or -WRITE file.</td>
</tr>
<tr>
<td>-CLOSE *</td>
<td>Closes all -READ and -WRITE files currently open.</td>
</tr>
<tr>
<td>-CMS</td>
<td>Executes a CMS command from within Dialogue Manager.</td>
</tr>
<tr>
<td>-CMS RUN</td>
<td>In CMS, loads and executes a user-written subroutine.</td>
</tr>
</tbody>
</table>
## Overview of Dialogue Manager Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-CRTCLEAR</td>
<td>Clears the screen display.</td>
</tr>
<tr>
<td>-CRTFORM</td>
<td>Initiates full-screen variable data entry.</td>
</tr>
<tr>
<td>-DEFAULT</td>
<td>Presets initial values for variable substitution.</td>
</tr>
<tr>
<td>-DEFAULTS</td>
<td></td>
</tr>
<tr>
<td>-EXIT</td>
<td>Executes stacked commands and returns to the FOCUS prompt.</td>
</tr>
<tr>
<td>-GOTO</td>
<td>Establishes an unconditional branch.</td>
</tr>
<tr>
<td>-HTMLFORM</td>
<td>For use with the Web Interface to FOCUS.</td>
</tr>
<tr>
<td>-IF</td>
<td>Tests and branches control based on test results.</td>
</tr>
<tr>
<td>-INCLUDE</td>
<td>Dynamically incorporates one procedure in another.</td>
</tr>
<tr>
<td>-label</td>
<td>Is a user-supplied name that identifies the target for -GOTO or -IF.</td>
</tr>
<tr>
<td>-MVS RUN</td>
<td>Same as -TSO RUN.</td>
</tr>
<tr>
<td>-PASS</td>
<td>Sets password directly.</td>
</tr>
<tr>
<td>-PROMPT</td>
<td>Types a prompt message on the screen and reads a reply.</td>
</tr>
<tr>
<td>-QUIT</td>
<td>Exits the procedure without executing it.</td>
</tr>
<tr>
<td>-READ</td>
<td>Reads records from a non-FOCUS file.</td>
</tr>
<tr>
<td>-REPEAT</td>
<td>Executes a loop.</td>
</tr>
<tr>
<td>-RUN</td>
<td>Executes all stacked FOCUS commands and returns to procedure for further processing.</td>
</tr>
<tr>
<td>-SET</td>
<td>Assigns a value to a variable.</td>
</tr>
<tr>
<td>-TSO RUN</td>
<td>In MVS/TSO, loads and executes a user-written subroutine.</td>
</tr>
<tr>
<td>-TYPE</td>
<td>Types informative message to screen or other output device.</td>
</tr>
</tbody>
</table>
Managing Applications With Dialogue Manager

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-$WINDOW</td>
<td>Invokes Window Painter, transferring control from the procedure to the specified window file.</td>
</tr>
<tr>
<td>-$WRITE</td>
<td>Writes a record to a non-FOCUS file.</td>
</tr>
<tr>
<td>-&quot;...&quot;</td>
<td>Brackets contents for -CRTFORM display line.</td>
</tr>
<tr>
<td>-$? SET SETCOMMAND &amp;myvar</td>
<td>Captures the setting of SETCOMMAND in &amp;myvar.</td>
</tr>
<tr>
<td>-$? &amp; [string]</td>
<td>Displays the values of currently defined amper variables.</td>
</tr>
</tbody>
</table>

Sending a Message to the User: -TYPE

The Dialogue Manager command -TYPE enables you to send informative messages to the screen while a procedure is processing. These messages serve a variety of functions. They can explain the purpose of the procedure, the results of computations or calculations, or preface prompts requesting information from the terminal. -TYPE triggers these messages.

Syntax

How to Send a Message to the User

The syntax is

-TYPE[+] text
-TYPE[0] text
-TYPE[1] text

where:

+ Suppresses the line feed following the printing of text.

0 Forces a line feed before the message text is displayed.

1 Forces a page eject before the message text is printed.

text

Is all succeeding text including variable values supplied on the same command line. It sends the text to the screen, followed by a line feed. It remains on screen until scrolled off or replaced by a new screen.

The options +, 0, and 1 are used to pass printer control characters to the output device and are particularly useful for character printers. Options + and 1 do not work on IBM 3270-type terminals. -TYPE sends the text to the terminal as soon as it is encountered in the processing of a procedure.
**Example**  

**Sending a Message to the Users**

The following is an example of the use of -TYPE:

```
- Version 1 6/30/85 SALERPT FOCEXEC
- TYPE This report calculates percentage of returns
  TABLE FILE SALES
  .
  .
  .
```

**Note:** The -TYPE message need not be enclosed in quotation marks, since FOCUS understands that -TYPE signals a following textual message. If you use quotation marks, they will appear along with the message. This differs from the use of TYPE in MODIFY, where quotation marks are used as delimiters and must enclose informative text.

---

**Controlling Execution: -RUN, -EXIT, and -QUIT**

Dialogue Manager enables you to manage the flow of execution with these commands:

- -RUN executes stacked commands and continues the procedure.
- -EXIT executes stacked commands and exits the procedure.
- -QUIT cancels execution and exits the procedure.

**Executing Stacked Commands and Continuing the Procedure: -RUN**

The Dialogue Manager command -RUN causes immediate execution of all stacked FOCUS commands and closes any external files opened with -READ or -WRITE. Following execution, processing of the procedure continues with the line that follows -RUN.
Example Executing Stacked Commands and Continuing the Procedure

The following example illustrates the use of -RUN to execute stacked code and then return to the procedure.

1. -TYPE This report calculates percentage of returns.
2. TABLE FILE SALES
   .
   .
   .
   END
3. -RUN
4. -TYPE This routine reports on data in the employee file.
   TABLE FILE EMPLOYEE
   .
   .
   .
   END

The procedure processes as follows:

1. The command -TYPE generates a message.
2. The FOCUS code is stacked.
3. The command -RUN causes the stacked commands to be executed and the output returned.
4. Processing continues with the line following -RUN. In this case, another message is sent and another TABLE request is initiated.

Executing Stacked Commands and Exiting the Procedure: -EXIT

-EXIT forces execution of stacked FOCUS commands as soon as it is encountered. However, instead of returning to the procedure, -EXIT closes all external files, terminates the procedure, and, either returns you to the FOCUS prompt or to the calling procedure.
### Executing Stacked Commands and Exiting the Procedure

In the following example, either the first TABLE request or the second TABLE request will execute, but not both:

1. `-TYPE` This report calculates percentage of returns.
2. `-IF &PROC EQ 'EMPLOYEE' GOTO EMPLOYEE;`
3. `-SALES
   
   TABLE FILE SALES
   .
   .
   .
   END`

4. `-EXIT
   -EMPLOYEE
   TABLE FILE EMPLOYEE
   .
   .
   .
   END`

The procedure processes as follows:

1. The command `-TYPE` generates a message.
2. Assume the value passed to `&PROC` is `SALES`.
   The `-IF` test checks the value of `&PROC`. Since it is not equal to `EMPLOYEE`, control passes to the label `-SALES`.
3. The FOCUS code is stacked. Control passes to the next line, `-EXIT`.
4. The command `-EXIT` executes the stacked commands. The output is sent to the terminal or output device and the procedure is exited.

   The TABLE request under the label `-EMPLOYEE` is not executed.

This example also illustrates an *implicit exit*. If the value of `&PROC` was `EMPLOYEE`, control would pass to the label `-EMPLOYEE` after the `-IF` test, and the procedure would never encounter the `-EXIT`. The TABLE FILE `EMPLOYEE` request would execute and the procedure would automatically terminate.
Canceling Execution of the Procedure: -QUIT

-QUIT cancels execution of any stacked commands and causes an immediate exit from the procedure.
This command is useful if tests or computations generate results that make additional processing unnecessary.

Example Canceling Execution of the Procedure

The following example illustrates the use of -QUIT to cancel execution based on the results of an -IF test.

1. -TYPE This report calculates percentage of returns.
   TABLE FILE SALES
   .
   .
   .
   END
2. -IF &CODE GT 'B10' OR &CODE EQ 'DONE' GOTO QUIT;
3. -QUIT

The procedure processes as follows:
1. The command -TYPE generates a message. The FOCUS code is stacked.
2. Assume that the value of &CODE is B11.
   The command -IF tests the value and passes control to -QUIT.
3. The command -QUIT cancels execution of the stacked commands and exits the procedure.

Exiting FOCUS and Setting Return Codes: -QUIT FOCUS

The Dialogue Manager command -QUIT FOCUS causes an immediate exit not only from the procedure, but from FOCUS as well. It returns you to the operating system and sets a return code.
### Syntax

**How to Exit FOCUS and Set a Return Code**

The syntax is

\`-QUIT FOCUS \[ n \mid 8 \]\`

where:

\`n \mid 8\`

Is the operating system return code number. It can be a constant or variable. A variable should be an integer. If you do not supply a value or if you supply a non-integer value, the return code posted to the operating system is 8 (the default value).

A major function of user-controlled return codes is to detect processing problems. The return code value determines whether to continue or terminate processing. This is particularly useful for batch processing.

### Branching

The execution flow of a procedure is determined with the following commands:

- **-GOTO.** Used for unconditional branching. `-GOTO` transfers control to a label.
- **-IF…GOTO.** Used for conditional branching. `-IF…GOTO` transfers control to a label depending on the outcome of a test.

### -GOTO Processing

Dialogue Manager processes a `-GOTO` as follows:

- It searches forward through the procedure for the target label. If it reaches the end without finding the label, it continues the search from the beginning of the procedure.
- The first time through a procedure, Dialogue Manager notes the addresses of all the labels so that they can be found immediately if needed again.
- Dialogue Manager takes no action on labels that do not have a corresponding `-GOTO`.
- If a `-GOTO` does not have a corresponding label, execution halts and an error message is displayed.
Syntax  How to Unconditionally Branch With -GOTO

The syntax is

-GOTO label
.
.
.
-GOTO label [TYPE text]

where:

label

Is a user-defined name of up to 12 characters. Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions or arithmetic or logical operations.

The label may precede or follow the -GOTO command in the procedure.

TYPE text

Optionally sends a message to a client application.

Example  Unconditional Branching With -GOTO

The following example “comments out” all the FOCUS code using an unconditional branch rather than -* in front of every line:

-GOTO DONE
TABLE FILE SALES
PRINT UNIT_SOLD RETURNS
WHERE PROD_CODE BETWEEN '&CODE1' AND '&CODE2'
AND PRODUCT = '&PRODUCT'
BY PROD_CODE,CITY
END
-RUN
-DONE
**Syntax**

**How to Conditionally Branch With -IF...GOTO**

The syntax is

```
-IF expression [THEN] GOTO label1; [ELSE IF...;]
[ELSE GOTO label2;]
```

where:

- **expression**
  - Is a valid expression. Literals need not be enclosed in single quotation marks unless they contain embedded blanks or commas.

- **THEN**
  - Is an optional keyword that increases readability of the command.

- **GOTO label**
  - Is a user-defined name of up to 12 characters. Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions or arithmetic or logical operations.
  - The label may precede or follow the -IF command in the procedure.

- **ELSE IF**
  - Optionally specifies a compound -IF test. See *Compound -IF Tests* on page 2-18.

- **ELSE GOTO**
  - Optionally passes control to label2 when the -IF test fails.

The command -IF must end with a semicolon (;) to signal that all logic has been specified. Continuation lines must begin with a hyphen (-).
Example  Conditional Branching With -IF...GOTO

In the following example, control passes to the label -PRODSALES if &OPTION is equal to S. Otherwise, control falls through to the label -PRODRETURNS, the line following the -IF test.

```
-IF &OPTION EQ 'S' GOTO PRODSALES;
-PRODRETURNS
    TABLE FILE SALES
    .
    .
    END
-EXIT
-PRODSALES
    TABLE FILE SALES
    .
    .
    END
-EXIT
```

The following command specifies both transfers explicitly:

```
-IF &OPTION EQ 'S' GOTO PRODSALES ELSE
    GOTO PRODRETURNS;
```

Notice that the continuation line begins with a hyphen (-).

Compound -IF Tests

You can use compound -IF tests provided each test specifies a target label.

Example  Using Compound -IF Tests

In the following example, if the value of &OPTION is neither R nor S, the procedure terminates (GOTO QUIT). The -QUIT serves both as a target label for the GOTO and as an executable command.

```
-IF &OPTION EQ 'R' THEN GOTO PRODRETURNS ELSE IF
    &OPTION EQ 'S' THEN GOTO PRODSALES ELSE
    GOTO QUIT;
    .
    .
-QUIT
```
Using Operators and Functions in -IF Tests

Expressions in an -IF test can include all FOCUS arithmetic and logical operators, as well as available functions or subroutines. See the Creating Reports manual for details.

Example Testing System and Statistical Variables

You can use system and statistical variables in -IF tests.

In the following example, if data (&LINES) is retrieved with the request, then the procedure branches to the label -PRODSALES; otherwise, it terminates.

```
TABLE FILE SALES
.
.
.
-IF &LINES NE 0 GOTO PRODSALES;
-EXIT
-PRODSALES
.
.
.
```

Screening Values With -IF Tests

To ensure that a supplied value is valid in a procedure, you can test it for the following:

- Presence
- Length
- Type

For instance, you would not want to perform a numerical computation on a variable for which alphanumeric data has been supplied.
Managing Applications With Dialogue Manager

**Syntax**  

**How to Test for the Presence of a Value**

The syntax is

```
-IF &name.EXIST [expression ]GOTO label...;
```

where:

- **&name**
  
  Is a user-supplied variable.

- **.EXIST**
  
  Indicates that you are testing for the presence of a value. If a value is not present, a zero (0) is passed to the expression. Otherwise, a non-zero value is passed.

- **expression**
  
  Is the remainder of a valid expression that uses &name.EXIST as an amper variable.

- **GOTO label**
  
  Specifies a label to branch to.

**Example**  

**Testing for the Presence of a Variable**

In the following example, if no value is supplied, &OPTION.EXIST is equal to zero and control is passed to the label -CANTRUN. The procedure sends a message to the client application and then exits. If a value is supplied, control passes to the label -PRODSALES.

```
-IF &OPTION.EXIST GOTO PRODSALES ELSE GOTO CANTRUN;

-PRODSALES
  TABLE FILE SALES
  .
  .
  .
END
-EXIT
-CANTRUN
-TYPE TOTAL REPORT CAN'T BE RUN WITHOUT AN OPTION.
-EXIT
```
Branching

Syntax  How to Test for the Length of a Value

The syntax is

-IF &name.LENGTH expression GOTO label...;

where:

&name
Is a user-supplied variable.

.LENGTH
Indicates that you are testing for the length of a value. If a value is not present, a zero (0) is passed to the expression. Otherwise, the number of characters in the value is passed.

expression
Is the remainder of a valid expression after &name is expanded.

GOTO label
Specifies a label to branch to.

Example  Testing for Variable Length

In the following example, if the length of &OPTION is greater than one, control passes to the label -FORMAT, which informs the client application that only a single character is allowed.

-IF &OPTION.LENGTH GT 1 GOTO FORMAT ELSE
-GOTO PRODSALES;
.
.
.
-PRODSALES
    TABLE FILE SALES
    .
    .
    .
END
-EXIT
-FORMAT
    -TYPE ONLY A SINGLE CHARACTER IS ALLOWED.

Example  Storing the Length of a Variable

The following example sets the variable &WORDLEN to the length of the string contained in the variable &WORD:

-PROMPT &WORD.ENTER WORD.
-SET &WORDLEN = &WORD.LENGTH;
**Syntax**

**How to Test for the Type of a Value**

The syntax is

```
-IF &name.TYPE expression GOTO label...;
```

where:

- **&name**
  
  Is a user-supplied variable.

- **.TYPE**
  
  Indicates that you are testing for the type of a value. The letter N (numeric) is passed to the expression if the value can be interpreted as a number up to $10^9$–1 and can be stored in four bytes as a floating point format. In Dialogue Manager, the result of an arithmetic operation with numeric fields is truncated to an integer after the whole result of an expression is calculated. If the value could not be interpreted as numeric, the letter A (alphanumeric) or the letter U (undefined) is passed to the expression.

- **expression**
  
  Is the remainder of a valid expression after &name is expanded.

- **GOTO label**
  
  Specifies a label to branch to.

**Example**

**Testing for Variable Type**

In the following example, if &OPTION is not alphanumeric, control passes to the label -NOALPHA, which informs the client application that only alphanumeric characters are allowed.

```
-IF &OPTION.TYPE NE A GOTO NOALPHA ELSE
-  GOTO PRODSALES;
  .
  .
-PRODSALES
  TABLE FILE SALES
  .
  .
  END
-EXIT
-NOALPHA
-TYPE ENTER A LETTER ONLY.
```
Testing the Status of a Query

The system variable &RETCODE returns a code after a query is executed. If the query results in a normal display, the value of &RETCODE is 0. If a display error occurs, or no display results (as can happen when the query finds no data), the value of &RETCODE is 8. (If the error occurs on a ? SU, the value of &RETCODE is 16.)

The value of &RETCODE is set following the execution of any of these queries:

<table>
<thead>
<tr>
<th>Query</th>
<th>NORMAL</th>
<th>NODISPLAY</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>? HOLD</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? SU*</td>
<td>0</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>? JOIN</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? COMBINE</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? DEFINE</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? USE</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? LOAD</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>? FILEDEF</td>
<td>0</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

*The &RETCODE value of ? SU means: 0 indicates that the FOCUS Database Server (formerly called the sink machine) is up with one or more users; 8 indicates that the FOCUS Database Server is up with no users; 16 indicates that there is an error in communicating to the FOCUS Database Server.

You can test the status of any of these queries by checking the &RETCODE variable and providing branching instructions in your procedure.

For example, if you are using Simultaneous Usage (SU), you must know if the FOCUS Database Server is available before you can begin a particular procedure. The following procedure tests whether SINK1 is available before launching PROC1.

? SU SINK1
-RUN
-IF &RETCODE EQ 16 GOTO BAD;
-INCLUDE PROC1
-BAD
-EXIT
Managing Applications With Dialogue Manager

Looping

The Dialogue Manager command -REPEAT allows looping in a procedure.

**Syntax**

**How to Specify a Loop**

The syntax is

-REPEAT label n TIMES
-REPEAT label WHILE condition
-REPEAT label FOR &variable [FROM fromval] [TO toval] [STEP s]

where:

*label*

Identifies the end of the code to be repeated (the loop). A label can include another loop if the label for the second loop has a different name from the first.

*n TIMES*

Specifies the number of times to execute the loop. The value of *n* can be a local variable, a global variable, or a constant. If it is a variable, it is evaluated only once, so the only way to end the loop early is with -QUIT or -EXIT (you cannot change the number of times to execute the loop) or to branch out of the loop.

*WHILE condition*

Specifies the condition under which to execute the loop. The condition is any logical expression that can be true or false. The loop is run if the condition is true.

*FOR &variable*

Is a variable that is tested at the start of each execution of the loop. It is compared with the value of *fromval* and *toval* (if supplied). The loop is executed only if *&variable* is less than or equal to *toval* (STEP is positive), or greater than or equal to *toval* (STEP is negative).

*FROM fromval*

Is a constant that is compared with *&variable* at the start of each execution of the loop. The default value is 1.

*TO toval*

Is a value that is compared with *&variable* at the start of each execution of the loop. The default is 1,000,000.

*STEP s*

Is a constant used to increment *&variable* at the end of each execution of the loop. It may be positive or negative. The default value is 1.

The parameters FROM, TO, and STEP can appear in any order.
Looping

Example  Using -REPEAT to Create a Loop

These examples illustrate how to write each of the syntactical elements of -REPEAT.

1. -REPEAT label n TIMES

Example:

-REPEAT LAB1 2 TIMES
-TYPE INSIDE
-LAB1 TYPE OUTSIDE

The output is:
INSIDE
INSIDE
OUTSIDE

2. -REPEAT label WHILE condition

Example:

-SET &A = 1;
-REPEAT LABEL WHILE &A LE 2;
-TYPE &A
-SET &A = &A + 1;
-LABEL TYPE END: &A

The output is:
1
2
END: 3

3. -REPEAT label FOR &variable FROM fromval TO toval STEP s

Example:

-REPEAT LABEL FOR &A STEP 2 TO 4
-TYPE INSIDE &A
-LABEL TYPE OUTSIDE &A

The output is:
INSIDE 1
INSIDE 3
OUTSIDE 5
Ending a Loop

A loop can end in one of three ways:

- It executes in its entirety.
- A -QUIT or -EXIT is issued.
- A -GOTO is issued to a label outside of the loop.

**Note:** If you later issue another -GOTO to return to the loop, the loop proceeds from the point it left off.

Using Expressions: -SET

The Dialogue Manager command -SET can be used in various ways to define values for variables in Dialogue Manager. You can compute new variables or recompute existing ones using arithmetic and logical expressions. You can also control loops, set indexes for variables, and call subroutines.

The following is a list of what can be included in a -SET expression and some specific rules for computations when using amper variables. Some calculations and special functions require that the amper variables have numeric values. FOCUS substitutes the value before placing the calculation in the stack. The variable does not have to have an I (integer) format, but the value for the variable must not contain alphanumeric characters. Note that the LAST operator used for reporting has no meaning in Dialogue Manager, nor do special MODIFY functions like FIND or LOOKUP.

- You can perform concatenations with the concatenation symbol. You must insert a space separating the amper variable from the concatenation symbol.
- You can use the DECODE function.
- You can use the EDIT function; however, its use is limited to the mask option.
- You can use the date functions.
- You can use subroutines.

For more information on expressions, functions, and subroutines, see the *Creating Reports* manual.
Computing a New Variable

You can use -SET to define a value for a substituted variable based on the results of a logical or arithmetic expression or a combination.

Syntax

How to Compute a New Variable

The syntax is

```
-SET &name = expression;
```

where:

- **&name**
  - Is a user-supplied variable which has its value assigned with the expression.

- **expression**
  - Is an expression following the rules outlined in the Creating Reports manual, but with limitations as defined in this section. The semicolon after the expression is required to terminate the -SET command.

Example

Altering a Variable Value

The following example demonstrates the use of -SET to alter variable values based on tests.

```
-START
-TYPE RETAIL PRICE ABOVE OR BELOW $1.00 IN THIS REPORT?
-PROMPT &CHOICE.ENTER A OR B.
-SET &REL = IF &CHOICE EQ A THEN 'GT' ELSE 'LT';
   TABLE FILE SALES
   PRINT PROD_CODE UNIT_SOLD RETAIL_PRICE
   BY STORE_CODE BY DATE
   IF RETAIL_PRICE &REL 1.00
END
```

In the example, the &CHOICE variable receives either A or B as the value supplied through -PROMPT. Assuming the user enters the letter A, -SET assigns the string value GT to &REL. Then, the value GT is passed to the &REL variable in the procedure, so that the expanded FOCUS command at execution time is:

```
IF RETAIL_PRICE GT 1.00
```

Note that literals are enclosed by single quotation marks. These are optional unless the literal contains embedded commas or blanks. To produce a literal that includes a single quotation mark, place two single quotation marks where you want one to appear.
Controlling a Loop With -SET

You can use the -SET statement to control the repetition limit of a loop.

Example Controlling a Loop With -SET

In the following example, the variable &N is incremented using -SET and tested to terminate the loop:

1. -DEFAULTS &N=0
2. -START
3. -SET &N=&N+1;
4. EX SLRPT
   -RUN
5. -IF &N GT 5 GOTO NOMORE;
6. -GOTO START
5. NOMORE TYPE EXCEEDING REPETITION LIMIT
   -EXIT

Execution proceeds in this way:

1. The -DEFAULTS statement initializes the loop-controlling variable &N to 0.
2. -START is a Dialogue Manager label that begins the loop. It is the target of an unconditional -GOTO.
3. The -SET statement increments the value of &N by one each time through the loop.
4. The FOCUS command EX SLRPT is stacked. The statement -RUN then calls for the execution of the stacked command.
5. This -IF statement tests the current value of the variable &N. If the value is greater than 5, control passes to the label -NOMORE, which displays a message for the user and forces an exit. If the value of &N is 5 or less, control falls through to the next Dialogue Manager statement.
6. The unconditional Dialogue Manager statement -GOTO START causes the loop to repeat.
Using the DECODE Function

You can use the DECODE function to change a variable to an associated value.

**Example** Assigning a Value to a Variable With DECODE

In the following example, the variable refers to a label:

1. `PROMPT &SELECT. ENTER CHOICE (A,B,C,D,E).
2. `SET &GO=DECODE &SELECT (A ONE B TWO C THREE
   -D FOUR E FIVE ELSE EXIT);
3. `GOTO &GO
   -ONE
   -TWO

The example processes as follows:

1. `PROMPT prompts the user at the terminal for a value for the variable &SELECT. Assume the user enters A.
2. `SET defines the variable &GO in terms of the DECODE function. Depending on the value input for &SELECT, DECODE associates a substitution. In this case, ONE is substituted for A.
3. `GOTO &GO transfers control to the label -ONE.

In the example, &GO can be another procedure (see Incorporating Multiple Procedures on page 2-35) that is executed, depending on the value that is decoded:

`TOP` `TYPE` `PROMPT &SELECT.ENTER 1, 2, 3, 4, 5, OR EXIT TO END.`
`SET &GO=DECODE &SELECT (1 ONE 2 TWO 3 THREE
   - 4 FOUR 5 FIVE ELSE EXIT);
`IF &GO IS EXIT GOTO EXIT;
`EX &GO` `RUN` `GOTO TOP` `EXIT`

For more information on DECODE, see the Creating Reports manual.
Using the EDIT Function

You can use the mask option of the EDIT function with amper variables. You can insert characters into an alphanumeric value, or extract certain characters from the value.

Example Using the EDIT Function With Amper Variables

In the following example, EDIT extracts a particular character, in this case the J, for comparison in order to branch to the appropriate label. Assume there are nested menus and the user must supply a number to branch to a particular menu. If the first character is a J, the branch is to the label JUMP which enables the user to jump in nested menus (the numbers refer to the explanation below):

1. TYPE CHOOSE 1 for Edit, 2 for Print, 3 for Math
2. TYPE TO JUMP LEVELS OF MENUS TYPE J1.3 ETC.
3. PROMPT &OPTION.A4. Please enter selection:.
4. SET &XYZ = EDIT(&OPTION, '9$$$')
5. IF &XYZ EQ J THEN GOTO JUMP;
   .
   .
6. JUMP
   .
   .

The example processes as follows:

1. TYPE send messages to the screen explaining the options to the user.
2. PROMPT asks the user to enter a value for the variable &OPTION. It can have as many as four characters.
3. SET calculates the variable &XYZ, which is the &OPTION variable, using the mask option of EDIT. The first character is screened.
4. IF determines the branch. If the variable &XYZ is equal to J, processing continues to the label JUMP. Otherwise, processing continues to the next command in the procedure.
5. JUMP is a label. The coding that follows contains the necessary FOCUS commands to enable the user to jump to the various menus.
Setting a Date

Natural date literals can be used in Dialogue Manager. They provide a way to take advantage of the powerful date handling capabilities of FOCUS. For more information on the FOCUS DATE format, see the Creating Reports manual.

Example Setting Dates and Computing the Difference in Days

Consider the following example:

-SET &NOW= 'MAR 11 1999';
-SET &LATER= '2000 11 MAR';
-SET &DELAY = &LATER - &NOW;

The value of &DELAY is set to the difference, in days, between &LATER and &NOW.

Note:

- A computation that adds or subtracts a fixed number of days from a variable in DATE format is not yet supported.
- A date given to Dialogue Manager cannot exceed 20 characters, including spaces.
- Dialogue Manager accepts only full-format dates (that is, MDY or MDYY, in any order).

Calling a Subroutine

Any function name encountered in a Dialogue Manager expression which is not recognized as a system standard name or FOCUS function is assumed to be a subroutine. These subroutines are externally programmed by users and stored in a library which is available at the time they are referenced. One or more arguments is passed to the user program, which performs an operation or calculation and returns a single value or character string.

Dialogue Manager variables can receive their values from subroutines through -SET.
Managing Applications With Dialogue Manager

**Syntax**

**How to Set a Variable Value Based on the Result From a Subroutine**

The syntax is

```
-SET &name = routine(argument,...,'format');
```

where:

- **name**
  - Is the name of the variable in which the result is stored.

- **routine**
  - Is the name of the subroutine.

- **argument**
  - Represents the argument(s) that must be passed to the subroutine. These arguments are converted to decimal format.

- **format**
  - Is the predefined format of the result. This is used to convert the numeric format back to character representation. It must be enclosed in single quotation marks.

**Example**

**Setting a Variable Value Based on the Result From a Subroutine**

In the following example, FOCUS invokes the subroutine RATE, adds 0.5 to the calculated value, and then formats the result as a double precision number. This result is then stored in the variable &COST:

```
-PROMPT &COMPANY.WHAT COMPANY ARE YOU USING?.
PROMPT &DEST.WHERE ARE YOU SENDING THE PACKAGE TO?.
PROMPT &WEIGHT.HOW HEAVY IS THE PACKAGE IN POUNDS?.
-SET &COST = RATE(&COMPANY,&DEST,&WEIGHT,'D6.2') + 0.5;
-TYPE THE COST TO SEND A &WEIGHT pound PACKAGE TO &DEST BY &COMPANY IS &COST
```

**Syntax**

**How to Load and Execute a Subroutine**

The following is an alternate way of calling subroutines. The Dialogue Manager command causes the subroutine to be loaded and then executed. The syntax is

```
{-CMS} RUN routine, argument,...
{-TSO} RUN routine, argument,...
{-MVS} RUN routine, argument,...
```

where:

- **routine**
  - Is the name of the subroutine.

- **argument**
  - Represents the argument(s) that must be passed to the subroutine. Arguments which are variables must have sizes predefined in prior -SET commands.
The numeric arguments to the subroutine are not automatically converted to D format in this syntax. Any required conversion must be done externally by the user or in the subroutine.

**Example**  Loading and Executing a Subroutine

The following is an example of the preceding syntax:

```
-PROMPT &MYCODE,A3.
-SET &MYNAME = '';
-SET &MYFACTOR = '';
-CMS RUN CODENAME, &MYCODE, &MYNAME, &MYFACTOR
```

In this example the program is CODENAME. The arguments that are variables are either prompted for or set at the beginning of the procedure and the values are then supplied for the arguments. Note that in this syntax the user program may use an argument for both input and output purposes. It is the responsibility of the user program to move the correct number of characters into the output variables.

**Additional Facilities**

Dialogue Manager supports a number of facilities for building applications. These facilities include:

- Creating startup files (profiles) that set overall environment conditions, which apply throughout your working session with FOCUS.
- Using -INCLUDE and EXEC to dynamically insert a procedure in another procedure, or to nest them up to four levels.
- Creating windows and menus for displaying information and collecting data in a procedure.
- Debugging procedures.
- Managing data integrity and security.
- Transferring data to and from non-FOCUS files.
Establishing Startup Conditions

FOCUS supports a startup profile that executes its content immediately upon entry into FOCUS. Using this procedure you can:

- Establish standard conditions that apply throughout the subsequent working session. For example, you can predefine environment parameters or automatically compute variables and make them available for later use.

- Provide a menu of subsequent user options.

- Control use of an application.

You can create a profile using any text editor or the FOCUS editor TED. The file is a FOCEXEC named PROFILE.

Note: It is possible to use an alternate FOCEXEC as a profile or not to execute a profile at all. For more information, see the Overview and Operating Environments manual.

Example: Creating a Startup Profile

Note the following example of a profile (under CMS):

```
USE
SALES FOCUS A1
MASTER FOCUS C1
END
CMS FILEDEF MYSAV DISK SAVE TEMP (LRECL 304 RECFM V
DEFINE FILE SALES
RATIO/D5.2 = (RETURNS/UNIT_SOLD);
END
-TYPE FOCUS SESSION ON &DATE MDYY &TOD
LET WORKREPORT=TABLE FILE EMPLOYEE
SET LINES=57, PAPER=66, PAGE=OFF
OFFLINE
```

Upon entering FOCUS, the profile is executed and a message showing the current date and time is displayed:

```
FOCUS SESSION ON 03/11/99 AT 14:21:06
```
Incorporating Multiple Procedures

Dialogue Manager supports dynamic inclusion of other procedures into a stored procedure at run time to enhance efficiency. There are two ways to do this:

- You can use the EXEC command in a procedure. The command will be stacked with other FOCUS commands and executed when an appropriate Dialogue Manager command forces execution of the stack. The procedure must be a fully executable procedure.

- The -INCLUDE command incorporates a file, which may be whole or partial procedures. A partial procedure could not be executed alone, but can be saved in a file and included in a calling procedure. This is particularly useful for procedures containing common header text, or partial processing cases that can be included at run time, based on tests and branches initiated in the original procedure. You can nest -INCLUDEs up to four levels.

The major difference between these two methods is when the procedure is executed. An EXEC command would be stacked and subsequently executed when the appropriate Dialogue Manager command is encountered, whereas -INCLUDE occurs immediately.

Using -INCLUDE

Lines inserted from a -INCLUDE are incorporated into the calling procedure as if they had originally been placed there.

There are many more uses for -INCLUDE files:

- As a control over the user environment. The included procedure must be present and some switches set before the present procedure continues execution.

- As a security mechanism. The included procedure can be encrypted and a direct password set. For more information, see the Describing Data manual.

- The name of the included file can be determined by the procedure (for example, -INCLUDE &NEWLINES, where NEWLINES is a variable whose value is a file name). This can shorten the main procedure when there are many alternate procedures.
Syntax  How to Incorporate a File

The syntax is

\[-\text{INCLUDE } \text{filename} \ [\text{filetype} \ [\text{filemode}]]\]

where:

filename
- Is the name of a FOCUS procedure.

filetype
- Is the procedure’s file type. If none is included, a file type of FOCEXEC is assumed.

filemode
- Is the procedure’s file mode. If none is included, a file mode of A is assumed.

Example  Incorporating a File

In this example, -INCLUDE searches for a file named DATERPT:

\[-\text{IF } \&\text{OPTION EQ S GOTO PRODSALES} \]
\[-\text{ELSE GOTO PRODRETURNS; } \]
\[-. \]
\[-. \]
\[-\text{PRODRETURNS} \]
\[-\text{INCLUDE DATERPT} \]
\[-\text{RUN} \]
\[-. \]
\[-. \]

Assume that DATERPT is a procedure containing the following TABLE request:

\[
\text{TABLE FILE SALES} \\
\text{PRINT PROD_CODE UNIT_SOLD} \\
\text{BY STORE_CODE} \\
\text{IF PROD_CODE IS } \&\text{PRODUCT} \\
\text{END}
\]

-INCLUDE incorporates this request into the calling procedure. FOCUS prompts for a value for the variable \&PRODUCT as soon as the -INCLUDE is encountered. The ensuing -RUN forces the execution of this included TABLE request.
**Example**  Incorporating Non-Executable Code

You can use `-INCLUDE` to call files containing code that is not executable. For instance, a common heading used throughout all reports can be stored in a separate file and incorporated into any procedure as needed. For example,

```plaintext
TABLE FILE SALES
-INCLUDE SALEHEAD
SUM UNIT_SOLD AND RETURNS AND COMPUTE …
```

where the SALEHEAD file contains:

```plaintext
HEADING
"THE ABC CORPORATION"
"RETAIL SALES DIVISION"
"MONTHLY SALES REPORT"
```

**Example**  Incorporating a Defined Field

As another example, a defined field can be placed in a separate file and called from a procedure as follows

```plaintext
-INCLUDE DEFRATIO
TABLE FILE SALES
-INCLUDE SALEHEAD
SUM UNIT_SOLD AND RETURNS AND RATIO
BY CITY
.
.
.
```

where the DEFRATIO file contains:

```plaintext
DEFINE FILE SALES
RATIO/D5.2=(RETURNS/UNIT_SOLD);
END
```

This DEFINE will be dynamically included before the TABLE request executes.
Nesting Procedures With -INCLUDE

Any number of different procedures can be invoked from a single calling procedure. You can also nest -INCLUDE commands within each other, up to four levels deep.

```
-PRODSALES
-INCLUDE FILE1
-RUN

FILE1
-INCLUDE FILE2
-RUN

FILE2
-INCLUDE FILE3
-RUN

FILE3
-INCLUDE FILE4
-RUN

FILE4
-RUN
```

Files 1 through 4 are incorporated into the original procedure. All of the included files are viewed as part of the original procedure. A procedure cannot branch to a label in an included file.

Using EXEC

A procedure can also call another one with the command EXEC (EX). The called procedure must be fully executable. You can also pass values to variables on the command line.

**Example** Using EXEC to Call a Procedure

In the following example, a procedure calls DATERPT:

```
-IF &OPTION EQ 'S' GOTO PRODSALES ELSE GOTO PRODRETURNS;
-PRODRETURNS
  EX DATERPT
  -RUN
```
Note: If the last executable statement in the called procedure is a -CRTFORM, control will not be returned to the calling procedure unless another Dialogue Manager command is included to terminate the -CRTFORM, such as -RUN or a -label.

Developing an Open-Ended Procedure

A file of stored FOCUS commands without variables looks and executes exactly as though it had been typed interactively into FOCUS from the terminal. However, if there is an error in your procedure file, it is rejected. If you make an error while typing interactively from the terminal, FOCUS issues prompts to help you correct the error.

If you store a procedure without the END command, you can execute all the procedure lines. The terminal then “opens” to allow interactive completion of the procedure. You can add additional command lines and enter the END command from the terminal to complete the procedure.

Note that you cannot use amper variables when typing online at a terminal. Open-ended procedures do not support variable substitution in lines entered after the terminal is opened. Variable substitution is supported in the stored portion of the procedure.

Example

Developing and Running an Open-Ended Procedure

Assume the following open-ended procedure is stored as SLRPT:

```
-TYPE ENTER REST OF PROCEDURE
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
```

You can invoke the procedure by typing EX SLRPT. It executes normally but fails to encounter an END command in the file. It then opens up the terminal displaying the FOCUS prompt. Depending on what you want, you could supply:

```
BY STORE_CODE
END
```

Or, alternatively:

```
IF CITY IS STAMFORD
BY STORE_CODE
END
```
Debugging With &ECHO and &STACK

It can be helpful to display command lines as they execute in order to test and debug procedures. The variable &ECHO is available for this purpose.

Syntax

How to Display Command Lines as They Execute

The syntax is

&ECHO = display

Valid values are:

ON
Displays lines that are expanded and stacked for execution.

ALL
Displays Dialogue Manager commands as well as lines that are expanded and stacked for execution.

OFF
Suppresses display of both stacked lines and Dialogue Manager commands. This value is the default.

You can set &ECHO through -DEFAULTS, -SET, or on the command line. For example, you can set ECHO to ALL for the execution of the procedure SLRPT using any of the following commands:

-DEFAULTS &ECHO = ALL
or
-SET &ECHO = ALL;
or
EX SLRPT ECHO = ON

If you use -SET or -DEFAULTS and place it in the procedure, display begins from that point in the procedure, and can be turned off and on again at any other point in the procedure.

Note that if the procedure is encrypted, &ECHO automatically receives the value OFF, regardless of the value that is assigned explicitly.

Testing Dialogue Manager Command Logic

To test the logic of Dialogue Manager commands, you can run the procedure but prevent actual execution of the stacked commands by setting the &STACK variable.
**Syntax**  **How to Test Dialogue Manager Command Logic**

The syntax is

\&STACK = \{ON | OFF\}

where:

**ON**
Results in normal execution of stacked commands. This value is the default.

**OFF**
Prevents execution of stacked commands. In addition, system variables (for example, \&RECORDS or \&LINES) are not set. Dialogue Manager commands are executed so you can test the logic of the procedure.

You can set \&STACK through -DEFAULTS, -SET, or on the command line. For example, you can set \&STACK to OFF for the execution of the procedure SLRPT using any of the following commands:

-DEFAULTS \&STACK = OFF

or

-SET \&STACK = OFF;

or

EX SLRPT STACK = OFF

This is usually used with ECHO = ALL for debugging purposes. The terminal displays both the Dialogue Manager commands, as well as the FOCUS commands with the supplied values. You can view the logic of the procedure.

**Locking Procedure Users Out of FOCUS**

Normally, users can respond to a Dialogue Manager value request with QUIT and return to the FOCUS command level or the prior procedure. In situations where it is important to prevent users from entering native FOCUS or QUIT from a particular procedure, the environment can be locked and QUIT deactivated.

**Syntax**  **How to Lock Procedure Users Out of FOCUS**

Enter the following command within the procedure:

-SET \&QUIT=OFF;

With QUIT deactivated, any attempt to return to native FOCUS produces an error message indicating that “quit” is not a valid value. Then the user is prompted for another value.
Managing Applications With Dialogue Manager

A user can terminate the FOCUS session from inside a locked procedure by responding to a prompt with

QUIT FOCUS

to return to the operating system, not the FOCUS command level.

Note: The default value for &QUIT is ON.

Writing to Files: -WRITE

In addition to conducting a dialogue with the user, Dialogue Manager can read from and write to files. For information on reading values from files, see Supplying Values Without Prompting on page 2-65.

The Dialogue Manager -WRITE enables you to write lines of text to a file.

Syntax  How to Write to a File

The syntax is

-WRITE ddname [NOCLOSE] text

where:

ddname

Is the logical name of the file as defined to FOCUS using FILEDEF, ALLOCATE, or DYNAM ALLOCATE. For information about file allocations, see the Overview and Operating Environments manual.

NOCLOSE

Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -READ statement is encountered.

text

Is any combination of variables and text. To write more than one line, end the first line with a comma (,) and begin the next line with a hyphen followed by a space (- ).

-WRITE opens the file to receiving the text and closes it upon exit from the procedure. When the file is reopened for writing, the new material overwrites the old. If you wish to reopen to add new records instead of overwriting existing ones, use the attribute DISP MOD when you define the file to the operating system.
**Example**  
**Writing to a File**

The following example reopens the file PASS under CMS to add new text:

- `CMS FILEDEF PASS DISK PASS DATA (DISP MOD
- WRITE PASS &DIV &RED &TEST RESULT IS,
- &RECORDS AT END OF RUN`

**Example**  
**Reading From and Writing to Sequential Files**

The following example illustrates reading from and writing to sequential files and the use of operating system commands (in this example, CMS). The numbers in the margin refer to notes that follow the example.

1. `TOP`
2. `PROMPT &CITY.ENTER NAME OF CITY -- TYPE QUIT WHEN DONE.`
3. `CMS FILEDEF PASS DISK PASS DATA A (LRECL 80 RECFM FB`
4. `WRITE PASS &CITY
   TABLE FILE SALES
   HEADING CENTER
   "LOWEST MONTHLY SALES FOR &CITY"
   " "
   PRINT DATE PROD_CODE
   BY LOWEST 1 UNIT_SOLD
   BY STORE_CODE
   BY CITY
   IF CITY EQ &CITY
   FOOTING CENTER
   "CALCULATED AS OF &DATE"
   ON TABLE SAVE AS INFO
END`
5. `RUN`
6. `CMS FILEDEF LOG DISK LOG DATA A1 (LRECL 80 RECFM FB`
   `MODIFY FILE SALES`
   `COMPUTE`
   `TODAY/I6=&YMD;`
   `CITY='&CITY';`
   `FIXFORM X5 STORE_CODE/A3 X15 DATE/A4 PROD_CODE/A3`
   `MATCH STORE_CODE DATE PROD_CODE`
   `ON MATCH TYPE ON LOG`
   `"<STORE_CODE><DATE><PROD_CODE><TODAY>"`
   `ON MATCH DELETE`
   `ON NOMATCH REJECT`
   `DATA ON INFO`
   `END`
7. `RUN`
   `EX SLRPT3`
8. `RUN`
11. `GOTO TOP`
12. `QUIT`
The procedure SLRPT3, which is invoked from the calling procedure, contains the following lines:

9. \texttt{-READ \&CITY.A8. TABLE FILE SALES HEADING CENTER \"MONTHLY REPORT FOR \&CITY\" \"LOWEST SALES DELETED\" = \" PRINT PROD_CODE UNIT_SOLD RETURNS DAMAGED BY STORE_CODE BY CITY IF CITY EQ \&CITY FOOTING CENTER \"CALCULATED AS OF \&DATE\" END}

10. \texttt{-RUN}

The following paragraphs explain the logic and show the dialogue between the user and the screen. User entries are in lowercase:

1. \texttt{-TOP} marks the beginning of the procedure.
2. \texttt{-PROMPT} sends the following prompt to the screen after the procedure is executed:

\begin{center}
\texttt{ENTER NAME OF CITY -- TYPE QUIT WHEN DONE>stamford}
\end{center}

3. \texttt{FILEDEF} defines and opens a file named PASS.
4. \texttt{-WRITE} writes the value of \&CITY to the non-FOCUS file named PASS. In this case the value written is STAMFORD.
5. -RUN executes the stacked TABLE request. In this case, a non-FOCUS file named INFO is created with the SAVE command. This is a sequential file, containing the result of the TABLE request as shown below.

<table>
<thead>
<tr>
<th>NUMBER OF RECORDS IN TABLE</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINES</td>
<td>7</td>
</tr>
<tr>
<td>(BEFORE TOTAL TESTS)</td>
<td></td>
</tr>
</tbody>
</table>

EBCDIC RECORD NAMED INFO

<table>
<thead>
<tr>
<th>FIELDNAME</th>
<th>ALIAS</th>
<th>FORMAT</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT_SOLD</td>
<td>SOLD</td>
<td>I5</td>
<td>5</td>
</tr>
<tr>
<td>STORE_CODE</td>
<td>SNO</td>
<td>A3</td>
<td>3</td>
</tr>
<tr>
<td>CITY</td>
<td>CTY</td>
<td>A15</td>
<td>15</td>
</tr>
<tr>
<td>DATE</td>
<td>DTE</td>
<td>A4MD</td>
<td>4</td>
</tr>
<tr>
<td>PROD_CODE</td>
<td>PCODE</td>
<td>A3</td>
<td>3</td>
</tr>
</tbody>
</table>

------------------------------------------
TOTAL                                       30
------------------------------------------

DEFAULT FILEDEF ISSUED

FILEDEF INFO DISK INFO FOCTEMP A1 (LRECL 30 BLKSIZE 300 RECFM F6)

6. FILEDEF defines a log file for the subsequent MODIFY request.

7. -RUN executes the stacked MODIFY request. The data comes directly from the INFO file created in the prior TABLE request and is entered using FIXFORM. Hence, the product with the lowest UNIT_SOLD is deleted from the file, and logged to a log file.

```
sales.foc ON 04/23/93 AT 12:28:38

transactions: total= 1 accepted= 1 rejected= 0
segments: input= 0 updated= 0 deleted= 1
```

8. The next -RUN executes another procedure called SLRPT3.

9. -READ reads the value for &CITY from the non-FOCUS file PASS. In this case the value passed is STAMFORD.
Managing Applications With Dialogue Manager

10. The -RUN executes the TABLE request and control is routed back to the calling procedure.

<table>
<thead>
<tr>
<th>STORE_CODE</th>
<th>CITY</th>
<th>PROD_CODE</th>
<th>UNIT_SOLD</th>
<th>RETURNS</th>
<th>DAMAGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>14B</td>
<td>STANFORD</td>
<td>B19</td>
<td>60</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B12</td>
<td>49</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B17</td>
<td>29</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C7</td>
<td>45</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D12</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E2</td>
<td>80</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E3</td>
<td>70</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

CALCULATING AS OF 04/23/93

11. -GOTO TOP routes control to the top.
12. When the user types QUIT, processing ends.

Using Variables in Procedures

Interactive variable substitution is at the heart of Dialogue Manager. You can create procedures that include variables (also called amper variables) and supply values for them at run time. These variables store a string of text or numbers and can be placed anywhere in a procedure. A variable can refer to a field, a command, descriptive text, a file name—literally anything.

Variables can be used only in procedures. They are ignored if you use them while creating reports live at the terminal. Values for variables may be supplied either directly on the command line when you execute the procedure, or through the -DEFAULTS command, the -SET command, or a -READ command in the procedure itself.

This section describes how to use amper variables in procedures and how to supply values for them. Variables fall into two classifications:

- Local and global variables have values supplied at run time. Local variable values remain in effect for the respective procedure, while global variable values remain in effect for all procedures executed during an entire FOCUS session (that is, from the time you enter FOCUS until you exit with the FIN command).
- System, statistical, and special variables have values that the system automatically resolves whenever you request them.
Leading double ampersands (&&) denote global variables. All other Dialogue Manager variables begin with a single ampersand (&). For this reason, in the FOCUS community they are known as amper variables.

The maximum number of local, global, system, statistical, special and index variables available in a procedure is 512. Approximately 30 are reserved for use by FOCUS.

Additionally, Dialogue Manager supports four types of prompting. You can alter the execution flow of your procedure, or change the substance of the request based on the values entered. These are:

- **Direct Prompting with -PROMPT**: You can request a set of values before they are actually needed. You can write your own text for these prompts and then validate the entered values to confirm that they fit a preset list of acceptable items or match a predefined format.

- **Full-Screen Data Entry with -CRTFORM**: The -CRTFORM command gathers variable values through full-screen data entry. Many values can be input and manipulated at the same time. Several screens can be included in a single procedure and used for a variety of purposes, including the development of menu-driven applications.

  -CRTFORM invokes FIDEL, the FOCUS Interactive Data Entry Language, and incorporates most of its functions. You can also use Screen Painter to design and paint -CRTFORM data entry screens directly on your terminal screen.

  Note that the Dialogue Manager command -CRTFORM is used for entering Dialogue Manager amper variable values. The equivalent MODIFY command, CRTFORM (without a hyphen), is used in MODIFY requests to enter field values.

- **Selecting Items from a Menu with -WINDOW**: You can create a series of menus and windows using the Window Painter facility and display them on the screen using the -WINDOW command. When displayed, the menus and windows can collect data by prompting users to select a value, enter a value, or press a program function (PF) key.

- **Implied Prompting**: FOCUS recognizes variables in a procedure by the leading ampersand (&). If a value has not been provided by some other means, FOCUS automatically requests a value from the terminal when needed.
Managing Applications With Dialogue Manager

Querying the Values of Variables

Amper variable values can be queried during execution.

**Syntax**

**How to Query the Values of Variables**

The syntax is

```
-? &[string]
```

where:

- **string**
  
  Is a complete amper variable or a partial string of up to 12 characters. Only amper variables starting with the specified string are displayed.

The command displays the following message, followed by a list of currently defined amper variables and their values:

```
CURRENTLY DEFINED & VARIABLES:
```

Note that this is a Dialogue Manager query. Since local variables do not exist outside a procedure, no similar query is available from the FOCUS command line.

Querying Parameter Value Settings

There is a Dialogue Manager query that enables you to capture previously defined SET parameter values in amper variables.

**Syntax**

**How to Query Parameter Value Settings**

The syntax is

```
-? SET parameter ampervar
```

where:

- **parameter**
  
  Is any valid FOCUS setting that may be queried with the ? SET or ? SET ALL command.

- **ampervar**
  
  Is the name of the variable where the value is to be stored.
Example: Querying a Parameter Value Setting

For example, if you enter

```bash
?- SET ASNAMES &abc
?- TYPE &ABC
```

the value stored in &abc becomes the value of ASNAMES. If you omit &abc from the command, then a variable called &ASNAMES is created that contains the value of ASNAMES.

Local Variables

Local variables are identified by a single ampersand (&) preceding the name of the variable. They remain in effect throughout a single procedure.

Example: Using Local Variables

In the following example, &CITY, &CODE1, and &CODE2 are local variables:

```plaintext
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
"PRODUCT CODES FROM &CODE1 TO &CODE2"
""
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
END
```

Assume you supply the values when you execute the procedure:

```
EX SLRPT CITY = STAMFORD, CODE1=B10, CODE2=B20
```

The procedure looks like this before it processes:

```plaintext
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR STAMFORD"
"PRODUCT CODES FROM B10 TO B20"
""
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ STAMFORD
BY PROD_CODE
IF PROD_CODE IS-FROM B10 TO B20
END
```
Values supplied for local variables remain current in the procedure. That is, all instances of the variables receive the values supplied. However, the values are not passed to other procedures containing the same variables (that is, &CODE1 and &CODE2 in another procedure). The values disappear after SLRPT has finished processing.

Global Variables

Global variables differ from local variables in that once a value is supplied, it remains current throughout the FOCUS session, unless set to another value with -SET or cleared by the LET CLEAR command. For information on LET CLEAR, see Chapter 5, Defining Word Substitutions: LET. They are useful for gathering values at the start of a work session for use by several subsequent procedures. All procedures which use a particular global variable will receive the current value until you exit from FOCUS.

Global variables are specified through the use of a double ampersand (&&) preceding the variable name. It is possible to have a local and global variable with the same name. They are distinct and may have different values.

Example Using Global Variables

The following is an example of a procedure containing global variables:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &&CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &&CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &&CODE1 TO &&CODE2
END
```

Syntax How to Query the Values of Global Variables

Since global variable values remain current throughout the FOCUS session, it is helpful to be able to display their values on demand. Do this by issuing the following command,

```
? &&
```

which displays the values of all global variables in use during the FOCUS session.
Example

Querying the Values of Global Variables

The following example displays the values of three global variables:

```
> ? &
  &CITY    STAMFORD
  &CODE1   B10
  &CODE2   B20
>
```

System Variables

FOCUS automatically substitutes values for system variables encountered in a Dialogue Manager request. System-supplied variables cannot be overridden. For example, you can use the system variable &DATE to automatically incorporate the system date in your request.

Reference

Summary of System Variables

A list of Dialogue Manager system variables follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE</td>
<td>MM/DD/YY</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;DATEfmt</td>
<td>Any date format.</td>
<td>Returns the current date, where fmt can be any combination of YYMD, MDYY, etc.</td>
</tr>
<tr>
<td>&amp;DMY</td>
<td>DDMYY</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;DMYY</td>
<td>DDM#CCYY</td>
<td>Returns the current (four-digit year) date.</td>
</tr>
<tr>
<td>&amp;FOCCPU</td>
<td>milliseconds</td>
<td>Calculates the OS CPU time. MVS only. In CMS, this returns the same value as &amp;FOCTTIME.</td>
</tr>
<tr>
<td>&amp;FOCEXTTRM</td>
<td>ON/OFF</td>
<td>Indicates the availability of extended terminal attributes.</td>
</tr>
</tbody>
</table>
### Variable: &FOCFIELDNAME

<table>
<thead>
<tr>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW</td>
<td>Returns a string indicating whether long and qualified field names are supported. A value of OLD means that they are not; NEW means that they are; and NOTRUNC means that they are supported, but unique truncations of field names cannot be used.</td>
</tr>
<tr>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td>NOTRUNC</td>
<td></td>
</tr>
</tbody>
</table>

### Variable: &FOCFOCSEXEC

Manages reporting operations involving many similarly named requests that are executed using EX. &FOCFOCSEXEC enables you to easily determine which procedure is running. &FOCFOCSEXEC can be specified within a request or in a Dialogue Manager command to display the name of the currently running procedure.

### Variable: &FOCINCLUDE

Manages reporting operations involving many similarly named requests that are included using -INCLUDE. &FOCINCLUDE can be specified within a request or in a Dialogue Manager command to display the name of the current included procedure.

### Variable: &FOCMODE

Identifies the operating environment.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMS</td>
<td></td>
</tr>
<tr>
<td>CRJE</td>
<td></td>
</tr>
<tr>
<td>MSO</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td></td>
</tr>
<tr>
<td>TSO</td>
<td></td>
</tr>
</tbody>
</table>

### Variable: &FOCPRINT

Returns the current print setting.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLINE</td>
<td></td>
</tr>
<tr>
<td>OFFLINE</td>
<td></td>
</tr>
</tbody>
</table>

### Variable: &FOCPUTLVL

(For example, 9306 or 9310.)

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCUS PUT level number.</td>
<td></td>
</tr>
</tbody>
</table>
### Using Variables in Procedures

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;FOCQUALCHAR</td>
<td>. : ! % \</td>
<td>Returns the character used to separate the components of qualified field names.</td>
</tr>
<tr>
<td>&amp;FOCREL</td>
<td>release number</td>
<td>Identifies the FOCUS Release number (for example, 6.5 or 6.8).</td>
</tr>
<tr>
<td>&amp;FOCSBORDER</td>
<td>ON OFF</td>
<td>Whether solid borders will be used in full-screen mode.</td>
</tr>
<tr>
<td>&amp;FOCSYSTYP</td>
<td>HIPER CP/A</td>
<td>CMS system type.</td>
</tr>
<tr>
<td>&amp;FOCTMPDSK</td>
<td>A ... Z</td>
<td>Identifies the disk where FOCUS places temporary work files (for example, HOLD files). CMS only.</td>
</tr>
<tr>
<td>&amp;FOCTRMSD</td>
<td>24 27 32 43</td>
<td>Indicates terminal height. (This can be any value; the examples shown are common settings.)</td>
</tr>
<tr>
<td>&amp;FOCTRMSW</td>
<td>80 132</td>
<td>Indicates terminal width. (This can be any value; the examples shown are common settings.)</td>
</tr>
<tr>
<td>&amp;FOCTRMTYP</td>
<td>3270 TTY UNKNOWN</td>
<td>Identifies the terminal type.</td>
</tr>
<tr>
<td>&amp;FOCTTIME</td>
<td>milliseconds</td>
<td>Calculates total CPU time. CMS only.</td>
</tr>
<tr>
<td>&amp;FOCVTIME</td>
<td>milliseconds</td>
<td>Calculates virtual CPU time. CMS only.</td>
</tr>
<tr>
<td>&amp;HIPERFOCUS</td>
<td>ON OFF</td>
<td>Returns a string showing whether HiperFOCUS is on.</td>
</tr>
</tbody>
</table>
Managing Applications With Dialogue Manager

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;IORETURN</td>
<td></td>
<td>Returns the code set by the last Dialogue Manager -READ or -WRITE operation.</td>
</tr>
<tr>
<td>&amp;MDY</td>
<td>MMDDYY</td>
<td>Returns the current date. The format make this variable useful for numerical comparisons.</td>
</tr>
<tr>
<td>&amp;MDYY</td>
<td>MMDDCCYY</td>
<td>Returns the current (four-digit year) date.</td>
</tr>
<tr>
<td>&amp;RETCODE</td>
<td>numeric</td>
<td>Returns the return code set upon execution of an operating system command. Executes all FOCUS commands in the FOCSTACK just as the -RUN statement would.</td>
</tr>
<tr>
<td>&amp;TOD</td>
<td>HH.MM.SS</td>
<td>Returns the current time. When you enter FOCUS, this variable is updated to the current system time only when you execute a MODIFY, SCAN, or FSCAN command. To obtain the exact time during any process, use the HHMMSS subroutine.</td>
</tr>
<tr>
<td>&amp;YMD</td>
<td>YYMDD</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;YYMD</td>
<td>CCYMMDD</td>
<td>Returns the current (four-digit year) date.</td>
</tr>
</tbody>
</table>

**Example**

**Using the System Variable &DATE**

The following example illustrates the use of a system variable in a request:

```
TABLE FILE SALES
.
.
.
FOOTING "CALCULATED AS OF &DATEMDYY"
END
-EXIT
```

The system variable &DATEMDYY ensures that the date that appears in the report is always the current system date.
Using the System Variable &FOCFOCEXEC

This next example illustrates how to use the system variable &FOCFOCEXEC in a request to display the name of the currently running procedure:

```
TABLE FILE EMPLOYEE
"REPORT: &FOCFOCEXEC -- EMPLOYEE SALARIES"
PRINT CURR_SAL BY EMP_ID
END
```

If the request is stored as a procedure called SALPRINT, when executed it will produce the following:

```
REPORT: DA0219 -- EMPLOYEE SALARIES
EMP_ID    CURR_SAL
---------  ---------
871382560  $11,080.00
112847512  $13,200.00
115360210  $6.00
117593129  $16,480.00
119265415  $5,500.00
119329144  $29,700.00
121495601  $6.00
123764317  $26,862.00
126724188  $21,120.00
219904371  $18,480.00
326179357  $21,780.00
451123478  $16,180.00
543729165  $9,080.00
818592173  $27,052.00
```

&FOCFOCEXEC and &FOCINCLUDE can also be used in -TYPE statements. For example, you have a procedure named EMPNAME which contains the following:

```
-TYPE &FOCFOCEXEC is: &FOCFOCEXEC
```

When EMPNAME is executed, the following output is produced:

```
&FOCFOCEXEC IS: EMPNAME
```
Managing Applications With Dialogue Manager

**Displaying a Date Variable Containing a Four-Digit Year**

You can display a date variable containing a 4-digit year without separators. The variables are &YYMD, &MDYY, and &DMYY. These variables complement the 2-digit year variables &YMD, &MDY, and &DMY.

**Example** Using the System Variable &YYMD

The following example shows a report using &YYMD:

```
TABLE FILE EMPLOYEE
HEADING
"SALARY REPORT RUN ON DATE &YYMD"
"  "
PRINT DEPARTMENT CURR_SAL
BY LAST_NAME BY FIRST_NAME
END
```

The resulting output for May 18, 1998 is:

```
PAGE 1
SALARY REPORT RUN ON DATE 19990319

<table>
<thead>
<tr>
<th>LAST_NAME</th>
<th>FIRST_NAME</th>
<th>DEPARTMENT</th>
<th>CURR_SAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANNING</td>
<td>JOHN</td>
<td>PRODUCTION</td>
<td>$29,708.00</td>
</tr>
<tr>
<td>BLACKWOOD</td>
<td>ROSEMARIE</td>
<td>MIS</td>
<td>$21,300.00</td>
</tr>
<tr>
<td>CROSS</td>
<td>BARBARA</td>
<td>OIS</td>
<td>$27,062.00</td>
</tr>
<tr>
<td>DAVIS</td>
<td>ELIZABETH</td>
<td>MIS</td>
<td>$0.00</td>
</tr>
<tr>
<td>GARDNER</td>
<td>DAVID</td>
<td>PRODUCTION</td>
<td>$0.00</td>
</tr>
<tr>
<td>GREENSPAN</td>
<td>MARY</td>
<td>MIS</td>
<td>$9,900.00</td>
</tr>
<tr>
<td>IRVING</td>
<td>JOHN</td>
<td>PRODUCTION</td>
<td>$26,062.00</td>
</tr>
<tr>
<td>JONES</td>
<td>DHINE</td>
<td>MIS</td>
<td>$18,400.00</td>
</tr>
<tr>
<td>MCCOY</td>
<td>JOHN</td>
<td>MIS</td>
<td>$18,400.00</td>
</tr>
<tr>
<td>MCKNIGHT</td>
<td>ROGER</td>
<td>PRODUCTION</td>
<td>$16,100.00</td>
</tr>
<tr>
<td>ROMANS</td>
<td>ANTHONY</td>
<td>PRODUCTION</td>
<td>$21,120.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>MARY</td>
<td>MIS</td>
<td>$13,200.00</td>
</tr>
<tr>
<td></td>
<td>RICHARD</td>
<td>PRODUCTION</td>
<td>$9,500.00</td>
</tr>
<tr>
<td>STEVENS</td>
<td>ALFRED</td>
<td>PRODUCTION</td>
<td>$11,000.00</td>
</tr>
</tbody>
</table>
```
Using Variables in Procedures

Statistical Variables

FOCUS posts many statistics concerning overall operations while a procedure executes in the form of statistical variables. As with system variables, FOCUS can automatically supply values for these variables on request.

Reference  Summary of Statistical Variables

A list of Dialogue Manager statistical variables follows:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ACCEPTS</td>
<td>Indicates the number of transactions accepted. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;BASEIO</td>
<td>Indicates the number of input/output operations performed.</td>
</tr>
<tr>
<td>&amp;CHNGD</td>
<td>Indicates the number of segments updated. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;DELTED</td>
<td>Indicates the number of segments deleted. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;DUPLS</td>
<td>Indicates the number of transactions rejected as a result of duplicate values in the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;FOCDISORG</td>
<td>Indicates the percentage of disorganization for a FOCUS file. This variable can be displayed or tested even if the value is less than 30% (the level at which ? FILE displays the amount of disorganization).</td>
</tr>
<tr>
<td>&amp;FOCERRNUM</td>
<td>Indicates the last error number, in the format FOCnnnn, displayed after the execution of a procedure. If more than one occurred, &amp;FOCERRNUM will hold the number of the most recent error. If no error occurred, &amp;FOCERRNUM will have a value of 0. This value can be passed to the operating system with the line -QUIT FOCUS &amp;FOCERRNUM. It can also be used to control branching from a procedure to execute an error-handling routine.</td>
</tr>
<tr>
<td>&amp;FORMAT</td>
<td>Indicates the number of transactions rejected as a result of a format error. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;INPUT</td>
<td>Indicates the number of segments added to the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;INVALID</td>
<td>Indicates the number of transactions rejected as a result of an invalid condition. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;LINES</td>
<td>Indicates the number of lines printed in last report. This variable applies only to report requests.</td>
</tr>
</tbody>
</table>
Managing Applications With Dialogue Manager

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;NOMATCH</td>
<td>Indicates the number of transactions rejected as a result of not matching a value in the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;READS</td>
<td>Indicates the number of records read from a non-FOCUS file.</td>
</tr>
<tr>
<td>&amp;RECORDS</td>
<td>Indicates the number of records retrieved in last report. This variable applies only to report requests.</td>
</tr>
<tr>
<td>&amp;REJECTS</td>
<td>Indicates the number of transactions rejected for reasons other than the ones specifically tracked by other statistical variables. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;TRANS</td>
<td>Indicates the number of transactions processed. This variable applies only to MODIFY requests.</td>
</tr>
</tbody>
</table>

**Example Using &LINES to Control Execution of a Request**

The following example illustrates how to use the statistical variable &LINES to control execution of a request:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/DS.2 - 100 * (RETURNS/UNIT_SOLD);
BY CITY
IF CITY EQ &CITY
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
ON TABLE HOLD
END
-RUN
-IF &LINES EQ 0 GOTO NORECORDS;
MODIFY FILE SALES
.
.
.
DATA ON HOLD
END
-RUN
-NORECORDS
-TYPE No record satisfies this report request
-QUIT
```

In the example, the system calculates the statistical variable &LINES (the number of lines produced by the TABLE request). If the number is 0, there are no lines in the report; -QUIT tells FOCUS to halt processing and the user is returned to the FOCUS prompt. If &LINES is greater than 0, processing continues to the MODIFY request.
Using Variables in Procedures

Syntax  How to Query the Values of Statistical Variables
You can query the current value of all statistical variables except &FOCDISORG and
&FOCERRNUM by typing the query command

? STAT
from the FOCUS prompt.

Special Variables
FOCUS provides special variables that apply to the cursor, function keys, windows, and other
features.

Reference  Summary of Special Variables
A list of special variables follow:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;CURSOR</td>
<td>Holds the cursor position.</td>
</tr>
<tr>
<td>&amp;CURSORAT</td>
<td>Reads the cursor position.</td>
</tr>
<tr>
<td>&amp;ECHO</td>
<td>Controls the display of commands for debugging purposes.</td>
</tr>
<tr>
<td>&amp;PFKEY</td>
<td>Holds the PF Key function.</td>
</tr>
<tr>
<td>&amp;QUIT</td>
<td>Controls whether the response QUIT, or PF1 in - CRTFORM, to a prompt causes an exit from the procedure.</td>
</tr>
<tr>
<td>&amp;STACK</td>
<td>Controls whether the entire procedure, or only the Dialogue Manager commands are executed.</td>
</tr>
<tr>
<td>&amp;WINDOWNAME</td>
<td>Holds the name of the last window activated by the most recently executed -WINDOW command (see Chapter 3, Designing Windows With Window Painter).</td>
</tr>
<tr>
<td>&amp;WINDOWVALUE</td>
<td>Holds the return value of the last window activated by the most recently executed -WINDOW command (see Chapter 3, Designing Windows With Window Painter).</td>
</tr>
</tbody>
</table>
Using Variables to Alter Commands

A variable can refer to a FOCUS command or to a particular field. In this way, the command structure of a procedure can be determined by the value of the variable.

Example Using a Field Variable

In the following example, the variable &FIELD determines the field to print in the TABLE request. For example, &FIELD could have the value RETURNS, DAMAGED, or UNIT_SOLD from a file named SALES.

```plaintext
TABLE FILE SALES
.
.
PRINT &FIELD
BY PROD_CODE
.
.
```

Evaluating a Variable Immediately

The .EVAL operator enables you to evaluate a variable’s value immediately, making it possible to change a procedure dynamically. It is used for substitution and re-evaluated by Dialogue Manager.

Syntax How to Evaluate a Variable

.EVAL uses the following syntax

```
[&]variable.EVAL
```

where:

```
variable
```

Is a local or global amper variable.

When the command procedure is executed, the expression is replaced with the value of the specified variable before any other action is performed.
Example Excluding and Including the .EVAL Operator

Without the .EVAL operator, an amper variable cannot be used in place of some FOCUS commands, as shown by the following example:

```
-SET &A='-TYPE';
&A HELLO
```

This example’s output shows that FOCUS does not recognize the value of &A:

UNKNOWN FOCUS COMMAND -TYPE

Appending the .EVAL operator to the &A amper variable makes it possible for FOCUS to interpret the variable correctly. For example, adding the .EVAL operator as follows,

```
-SET &A='-TYPE';
&A.EVAL HELLO
```

produces the following output:

HELLO

Example Evaluating a Variable Immediately

The .EVAL operator is particularly useful in modifying code at run time. The following example illustrates how to use the .EVAL operator in a record selection expression. The numbers to the left apply to the notes that follow:

1. -SET &R='IF COUNTRY IS ENGLAND';
2. -IF &Y EQ 'YES' THEN GOTO START;
3. -SET &R = '-*';
   -START
4. TABLE FILE CAR
   PRINT CAR BY COUNTRY
5. &R.EVAL
   END

The procedure executes as follows:

1. The procedure sets the value of &R to ‘IF COUNTRY IS ENGLAND’.
2. If the &Y is YES, the procedure branches to the START label, bypassing the second -SET statement.
3. If the &Y is NO, the procedure continues to the second -SET statement, which sets &R to ‘-*’, which is a comment.
4. The report request is stacked.
5. The procedure evaluates &R’s value. If the user wanted a record selection test, &R’s value is ‘IF COUNTRY IS ENGLAND’ and this line is stacked.

If the user did not want a record selection test, &R’s value is ‘-*’ and this line is ignored.
Concatenating Variables

You can append a variable to a character string or you can combine two or more variables and/or literals. See the Creating Reports manual for full details on concatenation. When using variables, it is important to separate each variable from the concatenation symbol (||) with a space.

Syntax How to Concatenate Variables

The syntax is

```
-SET &name3 = &name1 || &name2;
```

where:

- **&name3**
  - Is the name of the concatenated variable.

- **&name1 || &name2**
  - Are the variables, separated by a space and the concatenation symbol.

Note: The example shown uses strong concatenation, indicated by the || symbol. Strong concatenation removes any trailing blanks from &name1. Conversely, weak concatenation, indicated by the symbol |, preserves any trailing blanks in &name1.
Supplying Values for Variables at Run Time

When you design a Dialogue Manager procedure, you must decide how the variables in the procedure will acquire values. Values for variables can be supplied in two ways:

- When you call a procedure. You can include the variable names and their corresponding values as parameters in an EXEC command that calls one procedure from another.
- Directly in a procedure. The Dialogue Manager statements -DEFAULTS, -SET, and -READ enable you to supply values directly in a procedure.

**Example**

Supplying Values for Variables

The example in this section illustrates the use of the statements -DEFAULTS and -SET to supply values for variables. In the example, the user supplies the value of &CODE1, &CODE2, and &REGIONMGR as prompted by an HTML form.

The numbers to the left of the example apply to the notes that follow:

1. -DEFAULTS &VERB='SUM'
2. -SET &CITY=IF &CODE1 GT 'B09' THEN 'STAMFORD' ELSE 'UNIONDALE';
3. -TYPE REGIONAL MANAGER FOR &CITY
4. TABLE FILE SALES
   HEADING CENTER
   "MONTHLY REPORT FOR &CITY"
   "PRODUCT CODES FROM &CODE1 TO &CODE2"
   "="
   &VERB UNIT_SOLD AND RETURNS AND COMPUTE RATIO/D5.1 = 100 * (RETURNS/UNIT_SOLD);
   BY PROD_CODE
   IF PROD_CODE IS-FROM &CODE1 TO &CODE2
   FOOTING CENTER
5. "REGION MANAGER: &REGIONMGR"
   *CALCULATED AS OF &DATEMDYY*
   END
6. -RUN

The procedure executes as follows:

1. The -DEFAULTS statement sets the value of &VERB to SUM.
2. The -SET statement supplies the value for &CITY depending on the value for &CODE1 typed by the user on the form. Because the user typed B10 for &CODE1, the value for &CITY becomes STAMFORD.
3. When the user runs the report, FOCUS writes a message that incorporates the value for &CITY:

   REGIONAL MANAGER FOR STAMFORD
Managing Applications With Dialogue Manager

4. The user supplied the value for &REGIONMGR on the form. FOCUS supplies the current date at run time.

5. The FOCUS stack contains the following lines:

   TABLE FILE SALES
   HEADING CENTER
   "MONTHLY REPORT FOR STAMFORD"
   "PRODUCT CODES FROM B10 TO B20"
   SUM UNIT_SOLD AND RETURNS AND COMPUTE
   RATIO/D5.1 = 100 * (RETURNS/UNIT_SOLD);
   BY PROD_CODE
   IF PROD_CODE IS-FROM B10 TO B20
   FOOTING CENTER
   "REGION MANAGER: SMITH"
   "CALCULATED AS OF 03/11/99"
   END

Reference  General Rules for Supplying Variable Values

The following general rules apply to values for variables:

- The maximum length of a variable value to be displayed on the screen is 80 characters.
- A physical FOCSTACK line with values substituted for variables cannot exceed 80 characters; therefore, you should not use variable values longer than 80 characters.
- If a value contains an embedded space, comma (,) or equal sign (=), you must enclose the variable name in single quotation marks when you use it in an expression. For example, if the value for &CITY is NY, NY, you must refer to the variable as ‘&CITY’ in any expression.
- Once a value is supplied for a local variable, it is used throughout the procedure, unless it is changed by -CRTFORM, -PROMPT, -READ, -SET, or -WINDOW.
- Once a value is supplied for a global variable, it is used throughout the FOCUS session in all procedures, unless it is changed by -CRTFORM, -PROMPT, -READ, -SET, or -WINDOW, or cleared by LET CLEAR.
- Dialogue Manager automatically prompts the terminal if a value has not been supplied for a variable.
Supplying Values Without Prompting

There are several ways to supply values for local and global variables besides prompting methods. These are outlined below:

- Supplying values on the command line: You can supply values when you execute the procedure.
- Supplying values with -DEFAULTS: You supply initial default values in the procedure to ensure that you will not be implicitly prompted for the value.
- Supplying values with -SET: You supply values by setting them in the procedure using the -SET command. The values can be constants or the result of an expression.
- Supplying values with -READ: You can supply values by reading them in from a sequential file.

Supplying Values on the Command Line

When the user knows the values required by a procedure, they can be typed on the command line following the name of the procedure itself. This saves time, since FOCUS now has values to pass to each local or global variable and the user will not be prompted to supply them.

Example

Supplying Values on the Command Line

Consider the following procedure:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
BY CITY
IF CITY EQ &CITY
END
```

In order to execute this procedure and supply values for the variables on the command line, the user would type the following:

```
EX SLRPT CITY = STAMFORD, CODE1=B10, CODE2=B20
```
Managing Applications With Dialogue Manager

Syntax

How to Supply Values on the Command Line

Each name-value pair must have the syntactic form

\[ \text{name} = \text{value} \]

and pairs must be separated by commas. It is not necessary to enter the name-value pairs in the order that they are encountered in the procedure.

When the list of values to be supplied exceeds the width of the terminal, insert a comma as the last character on the line and enter the balance of the list on the following line(s), as shown:

```
EX SLRPT AREA=S, CITY = STAMFORD, VERB=COUNT, FIELDS = UNIT_SOLD, CODE1=B10, CODE2=B20
```

It is acceptable to supply some but not all values on the command line, in which case, values not supplied will trigger prompts to the terminal.

To supply global amper variable values on the command line, you must supply the double ampersand prefix, as in the following example:

```
EX SLRPT &&GLOBAL=value, CITY = STAMFORD, CODE1-B10, CODE2-B20
```

Using Positional Variables

When the variable is numbered (a positional variable; for example, &1, &2, &3) there is no need to specify the name, in this case a number, on the command line. FOCUS matches the values, one by one to the positional variables as they are encountered in the procedure. Therefore, it is vital to enter the appropriate value for each variable, in the proper order.

Example

Using Positional Variables

Consider the following example:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &1"
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
BY PROD_CODE
IF PROD_CODE IS-FROM &2 TO &3
BY CITY
IF CITY EQ &1
END

The command line for entry of positional values should read:

EX SLRPT STAMFORD, B10, B20
```
Supplying Values for Variables at Run Time

**Example**  Mixing Named and Positional Variables

You can mix named and positional variables freely on the command line, providing that names are associated with values for named variables and values are supplied for positional variables in the order that these variables are numbered in the procedure. For example:

```
EX SLRPT CITY = STAMFORD, B10, B20, VERB=COUNT
```

**Supplying Values With -DEFAULTS**

The Dialogue Manager command -DEFAULTS supplies an initial (default) value for a variable that had no value before the command was processed. It ensures that values will be passed to variables whether or not they are provided elsewhere.

**Syntax**  How to Supply Default Values

The syntax is

```
-DEFAULTS &{&}name=value [...] 
```

where:

- `&name`  
  Is the name of the variable.

- `value`  
  Is the default value assigned to the variable.

**Example**  Supplying Default Values

In the following example, -DEFAULTS sets default values for &CITY and &REGIONMGR.

```
-DEFAULTS &CITY=STAMFORD, &REGIONMGR=SMITH
TABLE FILE SALES
```

**Overriding Default Values**

You can override default values by supplying new values on the command line or by an explicit prompt.
Supplying Values With -SET

With -SET, you can assign a value computed in an expression.

Syntax

How to Set a Variable Value

The syntax is

-SET &name=expression;

where:

&name
Is the name of the variable.

expression;
Is a valid literal, arithmetic, or logical expression. Expressions can occupy several lines, so
you should end the command with a semicolon (;).

Example

Setting Variable Values

In the following example, -SET assigns the value 14Z or 14B to the variable &STORECODE, as
determined by the logical IF expression. The value of &CODE is supplied by the user.

-SET &STORECODE = IF &CODE GT C2 THEN '14Z' ELSE '14B';

TABLE FILE SALES
SUM UNIT_SOLD AND RETURNS
BY PROD_CODE
IF PROD_CODE GE &CODE
BY STORE_CODE
IF STORE_CODE IS &STORECODE
END

Example

Setting a Literal Value

Single quotation marks around a literal is optional unless it contains embedded blanks,
commas, or equal signs, in which case you must include them as shown:

-SET &NAME='JOHN O''HARA';

To assign a literal value that includes a single quotation mark, place two single quotation marks
where you want one to appear:

-SET &NAME='JOHN DOE';
Supplying Values for Variables at Run Time

Supplying Values With -READ

You can supply values for variables by reading them from a sequential file.

Syntax How to Supply Values With -READ

The syntax is

```
-READ ddname[,] [NOCLOSE] &name[.format.][,] ...
```

where:

**ddname**

Is the logical name of the file as defined to FOCUS using FILEDEF. (When using MVS, use ALLOCATE or DYNAM ALLOCATE.) A space after the ddname denotes a fixed format file while a comma denotes a comma-delimited file.

**NOCLOSE**

Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -WRITE statement is encountered.

**name**

Is the variable name. You may specify more than one variable. Using commas to separate variables is optional.

If the list of variables is longer than one line, end the first line with a comma and begin the next line with a dash followed by a blank (-). For example:

Comma-delimited files

```
-READ EXTFILE, &CITY, &CODE1,
   &CODE2
```

Fixed format files

```
-READ EXTFILE &CITY.A8, &CODE1.A3,,
```

**format**

Is the format of the variable. Note that format must be delimited by periods. The format is ignored for comma-delimited files.

**Note:** -SET provides an alternate method for defining the length of a variable using the corresponding number of characters enclosed in single quotation marks (‘). For example, the following command defines the length of &CITY as 8:

```
-SET &CITY=’ ’
```
Example  

Reading Data and Testing a System Variable

The example below reads data from EXTFILE, a fixed format file that contains the following data:

STAMFORDB10B20

The example tests the system variable &IORETURN. If there is no record to be read, the value of &IORETURN is not equal to zero and the procedure branches to the label after the TABLE request.

-IF &IORETURN NE 0 GOTO RESUME;
  TABLE FILE SALES
  SUM UNIT_SOLD
  BY CITY
  IF CITY IS &CITY
  BY PROD_CODE
  IF PROD_CODE IS-FROM &CODE1 TO &CODE2
  END
-RESUME
  .
  .
  .

Direct Prompting With -PROMPT

The Dialogue Manager command -PROMPT solicits values before the variables to which they refer are used in the procedure. The user is prompted for a value as soon as -PROMPT is encountered. If a looping condition is present, -PROMPT requests a new value for the variable, even if a value exists already. Thus, each time through the loop, the user is prompted for a new value.

With -PROMPT you can specify format, text, and lists in the same way as all other variables.
Supplying Values for Variables at Run Time

**Example**  
**Prompting for Variable Values**

The following is an example of the use of -PROMPT:

- PROMPT &CODE1
- PROMPT &CODE2
- SET &CITY = IF &CODE1 GT B09 THEN STAMFORD ELSE UNION;
- TYPE REGIONAL MANAGER FOR &CITY
- PROMPT &REGIONMGR
  
  TABLE FILE SALES
  HEADING CENTER
  "MONTHLY REPORT FOR &CITY"
  "PRODUCT CODES FROM &CODE1 TO &CODE2"
  SUM UNIT_SOLD AND RETURNS AND COMPUTE
  RATIO/D5.2 = 100 * (RETURNS/UNIT_SOLD);
  BY CITY
  IF CITY EQ &CITY
  BY PROD_CODE
  IF PROD_CODE IS-FROM &CODE1 TO &CODE2
  FOOTING CENTER
  "REGION MANAGER: &REGIONMGR"
  "CALCULATED AS OF &DATE"
  END

- PROMPT sends the following prompts to the screen. User input is shown in lowercase:

```
PLEASE SUPPLY VALUES REQUESTED
CODE1= h10
CODE2= h20
REGIONAL MANAGER FOR STAMFORD
REGIONMGR= smith
```

Note how the sequence of supplied values determines the overall flow of the procedure. The value of &CODE1 determines the value of &CITY which gives meaning to the -TYPE command. -TYPE gives the user the necessary information to make the correct choice when supplying the value for &REGIONMGR.

By default, all user input is automatically converted to uppercase.
Full-Screen Data Entry With -CRTFORM

-CRTFORM sets up full-screen menus for entering values. The -CRTFORM command in Dialogue Manager and the CRTFORM command in MODIFY are two versions of FIDEL for use in different contexts. The syntax, functions and features are fully outlined in the Maintaining Databases manual.

Selecting Data From Menus and Windows With -WINDOW

You can create a series of menus and windows using Window Painter, and then display those menus and windows on the screen using the -WINDOW command. When displayed, the menus and windows can collect data by prompting a user to select a value, to enter a value, or to press a program function (PF) key.

Implied Prompting

If a value is not supplied by any other means for a variable, FOCUS automatically prompts the user for the value. This is known as an implied prompt. These occur sequentially as each variable is encountered in the procedure.

Example: Automatically Prompting for Variable Values

Consider the following example:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY"
.
.
.
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1 TO &CODE2
.
.
.
FOOTING CENTER
"REGION MANAGER: &REGIONMGR"
"CALCULATED AS OF &DATE"
END
```
When you execute the procedure, FOCUS prompts for the values for the variables one at a
time. The terminal dialogue is as follows. User input is in lowercase:

```
PLEASE SUPPLY VALUES REQUESTED
CODE1=
   h10
CODE2=
   h20
REGIONMRG=
   smith
>
```

At the point when all variables have values, FOCUS processes the report request.

Verifying Input Values

Input values can be verified in the following ways:

- Format conditions can be specified against which the entered values are compared.
- Lists of acceptable values can be specified against which the entered values are compared.
- Text can be supplied that either explains what type of value is needed or lists choices of
  acceptable values on the screen.

Using Format Specifications

You can specify variables with format conditions against which the entered values are
compared. If the entered values do not have the specified format, FOCUS prints error messages
and prompts the user again for the value(s).

Alphanumeric formats are described by the letter A followed by the number of characters. The
number of characters can be from 1 to 255. Integer formats are described by the letter I
followed by the number of digits to be entered. The number can be from 1 to 9 (value must be
less than $2^{31}-1$).

The description of the format must be enclosed by periods.

If you test field names against input variable values, we recommend that you specify formats of
the input variables. If you do not, and the supplied value exceeds the format specification from
the Master File, the procedure is ended and error messages are displayed. To continue, the
procedure must be executed again. However, if you do include the format, and the supplied
value exceeds the format, the value is rejected by Dialogue Manager and the user is prompted
again.
Managing Applications With Dialogue Manager

Note: FOCUS internally stores all Dialogue Manager variables as alphanumeric codes. To perform arithmetic operations, Dialogue Manager converts the variable value to double-precision floating point decimal and then converts the result back to alphanumeric codes, dropping the decimal places. For this reason, do not perform tests that look for the decimal places in the numeric codes.

Example Using a Format Specification

Consider the following format specification:

&STORECODE.A3.

No special message is sent to the screen detailing the specified format. However, if, in the above example, the user enters more than three alphanumeric characters, the value is rejected, the error message FOC291 is displayed and the user is prompted again.

Note the following example detailing the dialogue between FOCUS and the user:

```
STORECODE => cc14
(FOC291) VALUE IN PROMPT REPLY EXCEEDS 03 CHAR: CC14
STORECODE =>
```

Using Lists of Value Ranges

Variables can be further customized by providing lists of values describing the acceptable range of prompted responses. If the user does not enter one of the available options, the terminal displays the list and re-prompts the user. This is an excellent way to limit the values supplied and to provide help information to the screen while prompting.

Example Providing a List of Valid Values

For example:

```
-PROMPT &CITY.(STAMFORD, UNIONDALE, NEWARK).
```

A message is printed if the user does not respond with one of the replies on the list. This is followed by a display of the value list. Finally, another prompt is issued for the needed value.

For example:

```
CITY> union
PLEASE CHOOSE ONE OF THE FOLLOWING:
   STAMFORD, UNIONDALE, NEWARK
CITY>
```
Supplying Values for Variables at Run Time

**Syntax**

**How to Use a Variable to Provide the Reply List**

You can also use a variable to provide the reply list, in conjunction with the -SET command. The syntax is

```
-SET &list='value,...';
-PROMPT &variable.(&list)[.text.]
```

where:

*list*  
Is the name of the reply list variable. Note that in the -PROMPT command, the value is substituted between the parentheses and delimited by periods. If the prompt text has parentheses, enclose that text in single quotation marks (‘).

*value*  
Is the desired value. You may list more than one value, separated by commas. Enclose the value(s) in single quotation marks (‘). A semicolon is required when using -SET.

*variable*  
Is the name of the variable for which you are prompting the user for values.

**Example**

**Using a Variable to Provide the Reply List**

For example:

```
-SET &CITIES='STAMFORD, UNIONDALE, NEWARK';
-PROMPT &CITY.(&CITIES).'(ENTER CITY)'.
```

The resulting screen is exactly the same as when the list itself is provided in the parentheses.

You can also create more complex combinations. For example:

```
-SET &CITIES=IF &CODE1 IS B10 THEN 'STAMFORD, NEWARK' ELSE 'STAMFORD, UNIONDALE, NEWARK';
```
Supplying Text for Variable Prompting

A variable can be further specified with customized text explaining the prompt at the screen.

**Example**

Supplying Text for Variable Prompting

For example:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR &CITY.ENTER CITY."
    
    
BY PROD_CODE
IF PROD_CODE IS-FROM &CODE1.A3.BEGINNING CODE. TO
    &CODE2.A3.ENDING CODE.
    
    
"REGION MANAGER: &REGIONMGR.REGIONAL SUPERVISOR."
"CALCULATED AS OF &DATEMDYY"
END
```

Notice that text has been specified for &CITY and &REGIONMGR without specification of a format.

Based on the example, the terminal displays the following prompts one by one:

```
Enter CITY> stamford
BEGINNING CODE> b10
ENDING CODE> b20
REGIONAL SUPERVISOR> smith
```
This section describes all the Dialogue Manager commands in alphabetical order. The following commands are included:

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>-*</td>
<td>-*</td>
<td>-CLOSE</td>
</tr>
<tr>
<td>-CLOSE *</td>
<td>-CMS</td>
<td>-CMS RUN</td>
</tr>
<tr>
<td>-CRTCLEAR</td>
<td>-CRTFORM</td>
<td>-DEFAULT</td>
</tr>
<tr>
<td>-DEFAULTS</td>
<td>-EXIT</td>
<td>-GOTO</td>
</tr>
<tr>
<td>-IF</td>
<td>-INCLUDE</td>
<td>-label</td>
</tr>
<tr>
<td>-MVS RUN</td>
<td>-PASS</td>
<td>-PROMPT</td>
</tr>
<tr>
<td>-QUIT</td>
<td>-READ</td>
<td>-REPEAT</td>
</tr>
<tr>
<td>-RUN</td>
<td>-SET</td>
<td>-TSO RUN</td>
</tr>
<tr>
<td>-TYPE</td>
<td>-WINDOW</td>
<td>-WRITE</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>&quot;&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Managing Applications With Dialogue Manager

Command:  

Syntax:  

where:

Is a comment. A space is not required between * and text.

Function:  
The command * signals the beginning of a comment line.

Any number of comment lines can follow one another, but each must begin with *. A comment line may be placed at the beginning or end of a procedure, or in between commands. However, it cannot be on the same line as a command.

Use comment lines liberally to document a procedure so that its purpose and history are clear to others.
**Command:** -?

**Syntax:**

```-? &\{string\}``

where:

- `string`
  
  Is an optional variable name of up to 12 characters. If this parameter is not specified, the current values of all local, global, and defined system and statistical variables are displayed.

**Function:** The command `-?` displays the current value of a local variable.
Managing Applications With Dialogue Manager

Command: -CLOSE

Syntax: -CLOSE ddname | *

where:

- **ddname**
  - Is the ddname of the open file described to FOCUS via an allocation (TSO, MSO) or FILEDEF (CMS) command.

- **:***
  - Closes all -READ and -WRITE files that are currently open.

Function: -CLOSE closes an external file opened with the -READ or -WRITE NOCLOSE option. The NOCLOSE option keeps a file open until the -READ or -WRITE operation is complete.
<table>
<thead>
<tr>
<th>Command:</th>
<th>-CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>CMS command where: command Is a CMS command.</td>
</tr>
<tr>
<td>Function:</td>
<td>CMS executes a CMS operating system command from within Dialogue Manager.</td>
</tr>
</tbody>
</table>
Managing Applications With Dialogue Manager

**Command:** -CMS RUN

**Syntax:**

```-CMS RUN subroutine```

where:

```
subroutine
```

Is a FOCUS user-written subroutine.

**Function:** In CMS, loads and executes the specified user-written subroutine.

**Similar Command:** SET can also execute user-written programs.
Command: -CRTCLEAR
Syntax: -CRTCLEAR
Function: Clears the current screen display.
Similar Command: None.
Managing Applications With Dialogue Manager

Command: -CRTFORM

Syntax: -CRTFORM [TYPE n] [BEGIN] [END [LOWER|UPPER]]

where:

-CRTFORM
Invokes FIDEL and signals the beginning of the screen form.

TYPE n
Enables you to define the number of lines (n) to reserve for messages.
You can specify a number from 1 to 4. The default is 4.

BEGIN
Supports the use of other Dialogue Manager commands to help build
the form.

END
Signals the end of the -CRTFORM. Used with -CRTFORM BEGIN.

LOWER
Reads lowercase data from the screen. Once you specify LOWER,
every screen thereafter is a lowercase screen until you specify
otherwise.

UPPER
Translates lowercase letters to uppercase. This is the default.

Function: Creates forms to prompt the user for values for variables.

All lines following a -CRTFORM command that begin with a hyphen and
enclose text in double quotation marks ("") are part of a single-screen form.
Pressing ENTER passes all input data to associated variables.

With -CRTFORM, the first line that does not begin with a -“ signals the
end of the form. With -CRTFORM BEGIN, the command -CRTFORM
END signals the end of the form.

All FIDEL facilities are available to -CRTFORM except HEIGHT,
WIDTH, and LINE.

Similar Command: CRTFORM in MODIFY functions identically to -CRTFORM in Dialogue
Manager.

See -PROMPT.
**Command:**  
-DEFAULT[S]

**Syntax:**  
-DEFAULTS &name=value, &name=value...

where:

=name  
Is the variable name.

=value  
Is the variable value.

**Function:** Sets initial values for the named variables in the procedure.

You can override -DEFAULTS values by supplying values for the variables on the command line, by specifically prompting for values with -PROMPT or -CRTFORM, or by supplying a value with -SET subsequent to -DEFAULTS.

-DEFAULTS guarantees that the variables are always given a value and therefore that it will execute correctly.

**Similar Command:** There is no similar command. However, default values are provided in other FOCUS modules to anticipate user needs and reduce the need for keystrokes in situations where most users desire a predefined outcome.

See also -SET.
Managing Applications With Dialogue Manager

Command: EXEC

Syntax: EX[EC] filename

where:

filename

Is the name of the procedure to be executed from within the current procedure.

Function: Specifies another procedure to be executed at run time. The specified procedure must be fully developed.

A procedure may contain more than one EXEC command. EXEC commands may be nested.

Note: Since EXEC is not a true Dialogue Manager dash command, it is stacked and executes only when -RUN, -EXIT, or the end of the procedure is encountered.

Similar Command: None.
Command:   -EXIT
Syntax:    -EXIT
Function:  -EXIT forces a procedure to end. All stacked commands are executed and
the procedure exits (if the procedure was called by another one, the calling
procedure continues processing).

Use -EXIT for terminating a procedure after processing a final branch that
completes the desired task.

The last line of a procedure is an implicit -EXIT. In other words, the
procedure ends after the last line is read.
**Managing Applications With Dialogue Manager**

**Command:** -GOTO

**Syntax:**

- GOTO label

  .

  .

  -label [TYPE text]

  where:

  *label*

  Is a user-defined name of up to 12 characters that specifies the target of the -GOTO action.

  Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic and logical operations, and so on.

  *TYPE text*

  Optionally sends a message to the client application.

**Function:**

- GOTO forces an unconditional branch to the specified label.

If Dialogue Manager finds the label, processing continues with the line following it.

If Dialogue Manager does not find the label, processing ends and an error message is displayed.
<table>
<thead>
<tr>
<th><strong>Command:</strong></th>
<th>-HTMLFORM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
<td>~HTMLFORM</td>
</tr>
<tr>
<td><strong>Function:</strong></td>
<td>For use with the Web Interface to FOCUS.</td>
</tr>
</tbody>
</table>
Command: -IF

Syntax:

```
-IF expression [THEN] GOTO label1; [ELSE GOTO label2;]
[ELSE IF...;]
```

where:

*label*

Is a user-defined name of up to 12 characters that specifies the target of the GOTO action.

Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic or logical operations, and so on.

*expression*

Is a valid expression. Literals need not be enclosed in single quotation marks unless they contain embedded blanks or commas.

*THEN*

Is an optional keyword that increases readability of the command.

*ELSE GOTO*

Optionally passes control to label2 when the -IF test fails.

*ELSE IF*

Optionally specifies a compound -IF test.

The semicolon (;) is required at the end of the command.

Continuation lines must begin with a hyphen (-).

Function:

-IF routes execution of a procedure based on the evaluation of the specified expression.

An -IF without an explicitly specified ELSE whose expression is false continues processing with the line immediately following it.
Command: -INCLUDE

Syntax: -INCLUDE filename [filetype [filemode]]

where:

filename
Is the procedure to be incorporated in the calling procedure.

filetype
Is the procedure’s file type. If none is included, a file type of FOCEXEC is assumed.

filemode
Is the procedure’s file mode. If none is included, a file mode of A is assumed.

Function: Specifies another procedure to be attached and executed at run time, as if it were part of the calling procedure. The specified procedure may comprise either a fully developed or partial procedure. Note that a partial procedure does not execute if called outside of the procedure containing -INCLUDE.

When using -INCLUDE, you may not branch to a label outside of the specified procedure.

A procedure may contain more than one -INCLUDE. Up to four -INCLUDEs may be nested.

You may use any valid command in a -INCLUDE.

Similar Command: EXEC, which may be used to execute a procedure inside another procedure.
Managing Applications With Dialogue Manager

**Command:**  
-label

**Syntax:**  
-label [TYPE message]

where:

-label  
Is a user-supplied name of up to 12 characters that identifies the target for a branch.

Do not use embedded blanks or the name of any other Dialogue Manager command except -QUIT or -EXIT. Do not use words that can be confused with functions, arithmetic or logical operations, and so on.

TYPE message  
Optionally sends a message to the client application.

**Function:**  
A label is the target of a -GOTO or -IF command.
Command: -MVS RUN
Syntax: -MVS RUN
Function: Same as -TSO RUN.
Managing Applications With Dialogue Manager

**Command:** -PASS

**Syntax:**

```
-PASS password
```

where:

- **password**
  
  Is a password or a variable containing a password.

**Function:**

Passwords can be directly issued and controlled by the Dialogue Manager. This is especially useful to specify a particular file or set of files which a given user can read or write. Passwords have detailed sets of functions associated with them through DBA module.

The procedure which sets passwords should be encrypted so that it and the passwords that it sets cannot be typed and made known.

A variable can be associated with -PASS so that a password value is prompted for and assigned.

**Similar Command:** The PASS command provides the same function at the command level as does the PASS parameter of the SET command.
Command: -PROMPT

Syntax: 
-PROMPT &name [[.format|(list)] [.text].]

where:

&name
Is a user-defined variable.

format
Optionally specifies alphanumeric or integer data type and length.

text
Optionally specifies prompting text that appears on the screen. Must be delimited by periods.

list
Optionally specifies a range of acceptable responses. Must be enclosed in parentheses.

Function: -PROMPT types a message to the terminal and reads the reply from the user. This reply assigns a value to the variable named.

If a format is specified and the supplied value does not conform, FOCUS displays an error message and prompts the user again for the value.

If a (list) is specified and the user does not reply with a value on the list, FOCUS reprompts and prints the list of acceptable values.

Note: You cannot use format and list together.

Similar Command: In MODIFY, PROMPT specifies additional data input needs.

In GRAPH, when it is set on, GPROMPT automatically prompts for all parameters needed to execute the graph request. This is quite a different function from -PROMPT in Dialogue Manager.

See -CRTFORM.
Managing Applications With Dialogue Manager

**Command:** -QUIT

**Syntax:**

-QUIT or -QUIT FOCUS \[n\]

where:

\[n\]

Is the operating system return code. It can be a constant or an integer variable up to 4095. If you do not supply a value or if you supply a non-integer value for \[n\], the return code is 8 (the default value).

**Function:** Forces an immediate exit from the procedure. Lines that have been stacked are not executed. This differs from an -EXIT, which executes all lines that are currently on the stack.

Like -EXIT, -QUIT returns the user to the FOCUS prompt.

-QUIT FOCUS takes the user out of FOCUS altogether and returns the user to the operating system level.

-QUIT can be made the target of a branch, with the same results as those already described.

**Similar Command:** QUIT can be entered in response to -PROMPT or -CRTFORM to force an exit from the procedure. The QUIT command can, however, be turned off from within Dialogue Manager to prevent the user from exiting FOCUS prompt.

The QUIT command can also be used to exit from MODIFY and TABLE requests as well as Dialogue Manager procedures.

The principle of QUIT remains consistent throughout FOCUS, namely that the exited request or procedure is not executed and the user is returned to the FOCUS prompt.

See also -RUN and -EXIT.
Command: -READ

Syntax: 
-READ ddname[,][NOCLOSE] &name[.format.][,]...

where:

ddname

Is the logical name of the file as defined to FOCUS using FILEDEF (or, for MVS, ALLOCATE or DYNAM ALLOCATE). A space after the ddname denotes a fixed format file while a comma denotes a comma-delimited file.

NOCLOSE

Indicates that the ddname should be kept open even after a -RUN is executed. The ddname is closed upon completion of the procedure or when a -CLOSE or subsequent -WRITE statement is encountered.

name

Is the variable name. You may specify more than one variable. Using a comma to separate variables is optional.

If the list of variables is longer than one line, end the first line with a comma and begin the next line with a dash followed by a blank (-) for comma-delimited files or a dash followed by a comma followed by a blank (,-) for fixed format files. For example:

Comma-delimited files

-READ EXTFILE, &CITY,&CODE1,
    - &CODE2

Fixed format files

-READ EXTFILE &CITY.A8. &CODE1.A3.,
    -, &CODE2.A3

format

Is the format of the variable. Note that format must be delimited by periods. The format is ignored for comma-delimited files.

Function: Reads data from non-FOCUS files. -READ can access data in either fixed or free form.

Similar Command: None.

See -WRITE.
Managing Applications With Dialogue Manager

**Command:**
-REPEAT

**Syntax:**

-REPEAT label n TIMES
-REPEAT label WHILE condition
-REPEAT label FOR &variable [FROM fromval] [TO toval] [STEP s]

where:

*label*

Identifies the code to be repeated (the loop). A label can include another loop if the label for the second loop has a different name from the first.

*n TIMES*

Specifies the number of times to execute the loop. The value of n can be a local variable, a global variable, or a constant. If it is a variable, it is evaluated only once, so the only way to end the loop early is with -QUIT or -EXIT (you cannot change the number of times to execute the loop).

*WHILE condition*

Specifies the condition under which to execute the loop. The condition is any logical expression that can be true or false. The loop is run if the condition is true.

*FOR &variable*

Is a variable that is tested at the start of each execution of the loop. It is compared with the value of fromval and toval (if supplied). The loop is executed only if &variable is less than or equal to toval (STEP is positive), or greater than or equal to toval (STEP is negative).

*FROM fromval*

Is a constant that is compared with &variable at the start of each execution of the loop. The default value is 1.

*TO toval*

Is a value against which &variable is tested. The default is 1,000,000.

*STEP s*

Is a constant used to increment &variable at the end of each execution of the loop. It may be positive or negative. The default value is 1.
Function: -REPEAT allows looping in a procedure.

The parameters FROM, TO, and STEP can appear in any order.

A loop ends when any of the following occurs:

- It is executed in its entirety.
- A -QUIT or -EXIT is issued.
- A -GOTO is issued to a label outside of the loop. If a -GOTO is later issued to return to the loop, the loop proceeds from the point it left off.
Managing Applications With Dialogue Manager

Command: -RUN

Syntax: -RUN

Function: -RUN causes immediate execution of all stacked FOCUS commands.

Following execution, processing of the procedure continues with the line that follows -RUN.

-RUN is commonly used to do the following:

• Generate results from a request that can then be used in testing and branching.

• Close an external file opened with -READ or -WRITE. When a file is closed, the line pointer is placed at the beginning of the file for a -READ. The line pointer for -WRITE is positioned depending on the allocation and definition of the file.
Command: \-SET

Syntax: \-SET \&name=expression;

where:

\&name

Is the name of a variable whose value will be set.

expression

Is a valid expression. Expressions can occupy several lines, so end the command with a semicolon (;).

Function: \-SET assigns a literal value to a variable, or a value that is computed in an arithmetic or logical expression.

Single quotation marks around a literal value are optional unless it contains embedded blanks or commas, in which case you must include them.
Managing Applications With Dialogue Manager

**Command:** -TSO RUN

**Syntax:**

- TSO RUN subroutine

where:

subroutine

Is the name of a FOCUS user-written subroutine.

**Function:** In TSO, loads and executes the specified user-written subroutine.

**Note:** The prefix -TSO can be used only with RUN.

**Similar Command:** -SET can also execute user-written programs.
Command: -TYPE

Syntax:
- TYPE[+] text
- TYPE[0] text
- TYPE[1] text

where:
- TYPE1
  Sends the text after issuing a page eject.
- TYPE0
  Sends the text after skipping a line.
- TYPE+
  Sends the text but does not add a line feed.

text
  Is a character string that fits on a line.

Function: Transmits informative messages to the user at the terminal. Any number of -TYPE lines may follow one another but each must begin with -TYPE.

Substitutable variables may be embedded in text. The values currently assigned to each variable will be displayed in their assigned position in the text.

- TYPE1 and TYPE+ are not supported by IBM 3270-type terminals.

Similar Command: TYPE is used in a variety of ways in FOCUS to send informative messages to the screen.

A TYPE command may appear on the same line as a label in Dialogue Manager.

In MODIFY, TYPE is used to print messages at the start and end of processes, at selected positions in MATCH or NOMATCH, NEXT or NONEXT, and to send a message after an INVALID data condition.
Managing Applications With Dialogue Manager

**Command:** -WINDOW

**Syntax:**

```-WINDOW windowfile winodowname [PFKEY|NOPFKEY] [GETHOLD] [BLANK|NOBLANK] [CLEAR|NOCLEAR]```

where:

- `windowfile` Identifies the file in which the windows are stored. In CMS, this is a file name. The file must have a file type of FMU. In MVS/TSO, this is a member name. The member must belong to a PDS allocated to ddname FMU.

- `winodowname` Identifies which window in the file will be displayed first.

- `PFKEY` Enables you to test for function key values during window execution.

- `NOPFKEY` You are unable to test for function key values during window execution.

- `GETHOLD` Retrieves stored amper variables collected from a Multi-Select window.

- `BLANK` Clears all previously set amper variable values when -WINDOW is encountered. This is the default setting.

- `NOBLANK` When -WINDOW is encountered, the values of previously set amper variables are retained.

- `CLEAR` Clears the screen before displaying the first window. This is the default behavior. When specified in conjunction with the Terminal Operator Environment (TOE), the TOE screen is redisplayed when control is transferred back to the procedure.

- `NOCLEAR` Displays the specified window directly over the current screen.
Function: Executes a window file. When the command is encountered, control is transferred from the procedure to the specified window file. The window specified in the command becomes the first active window. Control remains within the window file until a menu option is chosen, or a window is activated, for which there is no goto value.

The window file, and the windows in it, are created using Window Painter.
Managing Applications With Dialogue Manager

Command: -WRITE

Syntax: 
-WRITE ddname [NOCLOSE] text

where:

ddname
Is the logical name of the file as defined to FOCUS using FILEDEF (or for MVS, ALLOCATE or DYNAM ALLOCATE).

NOCLOSE
Indicates that the file should be kept open even if a -RUN is encountered. The file is closed upon completion of the procedure or when a -CLOSE or subsequent -READ statement is encountered.

text
Is any combination of variables and text. To write more than one line, end the first line with a comma (,) and begin the next line with a hyphen followed by a space (-).

Function: Writes information to non-FOCUS files.

Note that all files which have been written should be closed upon any exit from the procedure using -QUIT, -EXIT, or -RUN.

Similar Command: In TABLE, WRITE is a synonym for SUM; functionally it is quite different from -WRITE.

See -READ.
Command: 

```
-"
```

Syntax: 

```
- " 
```

where:

```
 " 
```

Enclose textual information, fields and spot markers.

Function: 
The -" " syntax is associated with the FIDEL -CRTFORM command. All textual data enclosed by the double quotation marks is printed to the screen. You can use position markers and specify variable fields within double quotation marks.

When -CRTFORM is processed, the screen displays a form and the cursor stops at each amper variable date entry field. If a variable has not been declared prior to the -CRTFORM, FOCUS prompts the user for a value to assign to the variable.

Similar Command: In MODIFY, enclosing data in double quotation marks (" ") without the leading hyphen is used with CRTFORM, or for headings, footings, subheads, and subfoots within a TABLE request.

See -CRTFORM.
Managing Applications With Dialogue Manager

System Defaults and Limits

This section provides you with an easier way of locating default values, operating system and FOCUS limits, summary tables, general rules, and tips for ease-of-use.

Some general rules to follow when you are creating procedures are:

- If a Dialogue Manager command exceeds one line, the following line must begin with a hyphen (-).
- The hyphen (-) must be placed at the first position of the command line.
- The command is usually attached to the hyphen (-), but you may leave space between the hyphen and the Dialogue Manager command.
- At least one space must be inserted between the Dialogue Manager command and other text.
- Procedure files must have the record format (RECFM) F and the logical record length (LRECL) 80.

The following are some general rules that apply in regard to supplying values for variables:

- The maximum length of a variable value is 79 characters.
- A physical FOCSTACK line with all variables expanded to their full values cannot exceed 80 characters. Since most variables are part of a line in a procedure, it is recommended that you use values that are less than 80 characters long.
- If a value contains an embedded comma (,) or embedded equal sign (=) the value must be enclosed between single quotation marks. For example:
  
  `EX SLRPT AREA=S, CITY='NY, NY'`

- Once a value is supplied for a local variable, it is used for that variable throughout the procedure, unless it is changed through a -PROMPT, -SET, or -READ.
- Once a value is supplied for a global variable, it is used for that global variable throughout the FOCUS session in all procedures, unless it is changed through a -PROMPT, -SET, or -READ.
- Dialogue Manager automatically sends a prompt to the terminal if a value has not been supplied for a variable. Automatic prompts (implied prompting) are identical in syntax and function to the direct prompts created with -PROMPT.
The following is a list of operating system default values, limits, and format specifications.

- The default value for the operating system return code value is 8.
- The maximum number of amper variables available in a procedure is 512, of which approximately 30 are reserved for use by FOCUS. This includes all local, global, system, statistical, special, and index variables.
- Literals must be surrounded by single quotation marks if they contain embedded blanks or commas. To produce a literal that includes a single quotation mark, place two single quotation marks where you want one to appear.
- Alphanumeric formats are described by the letter A followed by the number of characters. The number of characters can be from 1 to 255.
- Integer formats are described by the letter I followed by the number of digits to be entered. The number can be from one to nine digits in length, value must be less than $2^{31}-1$.
- A label is a user-defined name of up to 12 characters. You cannot use blanks and should not use the name of any other Dialogue Manager command. The label may precede or follow GOTO in the procedure.
- A date given to the Dialogue Manager cannot be more than 20 characters long, including spaces.
- -INCLUDE files can be nested up to 4 levels deep.
- The default setting for &QUIT is ON.
- When using Window Painter:
  - Screens should not begin in row 0, column 0, or column 1.
  - The maximum screen size is 22 rows by 77 columns.
  - A File Contents window has a limit of 12K worth of data. This is approximately 150 lines.
  - The maximum number of menu items is 41.
  - File Name windows must have a WIDTH of 24 or greater, or meaningless characters will appear.
The following tables summarize system, statistical, and special variables available in Dialogue Manager.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;DATE</td>
<td>%m/%d/%y</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;DATEfmt</td>
<td>Any combination of YYMD, MDYY, etc.</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;DMY</td>
<td>%d%m%y</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;DMYY</td>
<td>%d%m%m%y</td>
<td>Returns the current (four-digit year) date.</td>
</tr>
<tr>
<td>&amp;FOCCPU</td>
<td>milliseconds</td>
<td>Calculates the OS CPU time. MVS only. In CMS, this returns the same value as &amp;FOCTIME.</td>
</tr>
<tr>
<td>&amp;FOCEXTRM</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>&amp;FOCFIELDNAME</td>
<td>NEW</td>
<td>OLD</td>
</tr>
<tr>
<td>&amp;FOCFOCEXEC</td>
<td></td>
<td>Manages reporting operations involving many similarly named requests that are executed using EX. &amp;FOCFOCEXEC enables you to easily determine which procedure is running. &amp;FOCFOCEXEC can be specified within a request or in a Dialogue Manager command to display the name of the currently running procedure.</td>
</tr>
<tr>
<td>&amp;FOCINCLUDE</td>
<td></td>
<td>Manages reporting operations involving many similarly named requests that are included using -INCLUDE. &amp;FOCINCLUDE can be specified within a request or in a Dialogue Manager command to display the name of the current included procedure.</td>
</tr>
<tr>
<td>Variable</td>
<td>Format or Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&amp;FOCMODE</td>
<td>CMS CRJE MSO OS TSO</td>
<td>Identifies the operating environment.</td>
</tr>
<tr>
<td>&amp;FOCPRT</td>
<td>ONLINE OFFLINE</td>
<td>Returns the current print setting.</td>
</tr>
<tr>
<td>&amp;FOCPULVL</td>
<td>FOCUS PUT level number.</td>
<td>(For example, 9306 or 9310.)</td>
</tr>
<tr>
<td>&amp;FOCQUALCHAR</td>
<td>. : ! %</td>
<td>\</td>
</tr>
<tr>
<td>&amp;FOCREL</td>
<td>release number</td>
<td>Identifies the FOCUS Release number (for example, 6.5 or 6.8).</td>
</tr>
<tr>
<td>&amp;FOCSBORDER</td>
<td>ON OFF</td>
<td>Whether solid borders will be used in full-screen mode.</td>
</tr>
<tr>
<td>&amp;FOCSYSTYP</td>
<td>HIPER CP/A</td>
<td>CMS system type.</td>
</tr>
<tr>
<td>&amp;FOCTMPSK</td>
<td>A … Z</td>
<td>Identifies the disk where FOCUS places temporary work files (for example, HOLD files).</td>
</tr>
<tr>
<td>&amp;FOCTRMSD</td>
<td>24 27 32 43</td>
<td>Indicates terminal height. (This can be any value; the examples shown are common settings.)</td>
</tr>
<tr>
<td>&amp;FOCTRMSW</td>
<td>80 132</td>
<td>Indicates terminal width. (This can be any value; the examples shown are common settings.)</td>
</tr>
<tr>
<td>&amp;FOCTRMTYP</td>
<td>3270 TTY UNKNOWN</td>
<td>Identifies the terminal type.</td>
</tr>
<tr>
<td>&amp;FOCTTIME</td>
<td>milliseconds</td>
<td>Calculates total CPU time. CMS only.</td>
</tr>
</tbody>
</table>
### Managing Applications With Dialogue Manager

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format or Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;FOCVTIME</td>
<td>milliseconds</td>
<td>Calculates virtual CPU time. CMS only.</td>
</tr>
<tr>
<td>&amp;HIPERFOCUS</td>
<td>ON/OFF</td>
<td>Returns a string showing whether HiperFOCUS is on.</td>
</tr>
<tr>
<td>&amp;IORETURN</td>
<td></td>
<td>Returns the code set by the last Dialogue Manager -READ or -WRITE operation.</td>
</tr>
<tr>
<td>&amp;MDY</td>
<td>MMDDYY</td>
<td>Returns the current date. The format makes this variable useful for numerical comparisons.</td>
</tr>
<tr>
<td>&amp;MDYY</td>
<td>MMDDCCYY</td>
<td>Returns the current (four-digit) year.</td>
</tr>
<tr>
<td>&amp;RETCODE</td>
<td>numeric</td>
<td>Returns the return code set upon execution of an operating system command. Executes all FOCUS commands in the FOCSTACK just as the -RUN statement would.</td>
</tr>
<tr>
<td>&amp;TOD</td>
<td>HH.MM.SS</td>
<td>Returns the current time. When you enter FOCUS, this variable is updated to the current system time only when you execute a MODIFY, SCAN, or FSCAN command. To obtain the exact time during any process, use the HHMMSS subroutine.</td>
</tr>
<tr>
<td>&amp;YMD</td>
<td>YYMMDD</td>
<td>Returns the current date.</td>
</tr>
<tr>
<td>&amp;YYMD</td>
<td>CCYIMMDD</td>
<td>Returns the current (four-digit year) date.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;ACCEPTS</td>
<td>Indicates the number of transactions accepted. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;BASEIO</td>
<td>Indicates the number of input/output operations performed.</td>
</tr>
<tr>
<td>&amp;CHNGD</td>
<td>Indicates the number of segments updated. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;DELTD</td>
<td>Indicates the number of segments deleted. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td><strong>Variable</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&amp;DUPLS</td>
<td>Indicates the number of transactions rejected as a result of duplicate values in the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;FOCDISORG</td>
<td>Indicates the percentage of disorganization for a FOCUS file. This variable can be displayed or tested even if the value is less than 30% (the level at which ? FILE displays the amount of disorganization).</td>
</tr>
<tr>
<td>&amp;FOCERNUM</td>
<td>Indicates the last error number, in the format FOCnnnn, displayed after the execution of a procedure. If more than one occurred, &amp;FOCERNUM will hold the number of the most recent error. If no error occurred, &amp;FOCERNUM will have a value of 0. This value can be passed to the operating system with the line -QUIT FOCUS &amp;FOCERNUM. It can also be used to control branching from a procedure to execute an error-handling routine.</td>
</tr>
<tr>
<td>&amp;FORMAT</td>
<td>Indicates the number of transactions rejected as a result of a format error. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;INPUT</td>
<td>Indicates the number of segments added to the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;INVALID</td>
<td>Indicates the number of transactions rejected as a result of an invalid condition. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;LINES</td>
<td>Indicates the number of lines printed in last report. This variable applies only to report requests.</td>
</tr>
<tr>
<td>&amp;NOMATCH</td>
<td>Indicates the number of transactions rejected as a result of not matching a value in the data source. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;READS</td>
<td>Indicates the number of records read from a non-FOCUS file.</td>
</tr>
<tr>
<td>&amp;RECORDS</td>
<td>Indicates the number of records retrieved in last report. This variable applies only to report requests.</td>
</tr>
<tr>
<td>&amp;REJECTS</td>
<td>Indicates the number of transactions rejected for reasons other than the ones specifically tracked by other statistical variables. This variable applies only to MODIFY requests.</td>
</tr>
<tr>
<td>&amp;TRANS</td>
<td>Indicates the number of transactions processed. This variable applies only to MODIFY requests.</td>
</tr>
</tbody>
</table>
### Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;CURSOR</td>
<td>Holds the cursor position.</td>
</tr>
<tr>
<td>&amp;CURSORAT</td>
<td>Reads the cursor position.</td>
</tr>
<tr>
<td>&amp;ECHO</td>
<td>Controls the display of commands for debugging purposes.</td>
</tr>
<tr>
<td>&amp;PFKEY</td>
<td>Holds the PF Key function.</td>
</tr>
<tr>
<td>&amp;QUIT</td>
<td>Controls whether the response QUIT, or PF1 in -CRTFORM, to a prompt causes an exit from the procedure.</td>
</tr>
<tr>
<td>&amp;STACK</td>
<td>Controls whether the entire procedure, or only the Dialogue Manager commands are executed.</td>
</tr>
<tr>
<td>&amp;WINDOWNAME</td>
<td>Holds the name of the last window activated by the most recently executed -WINDOW command (see Chapter 3, Designing Windows With Window Painter).</td>
</tr>
<tr>
<td>&amp;WINDOWVALUE</td>
<td>Holds the return value of the last window activated by the most recently executed -WINDOW command (see Chapter 3, Designing Windows With Window Painter).</td>
</tr>
</tbody>
</table>
FOCUS Window Painter is a tool that helps you design and create your own menus and screens for attractive and easy-to-use applications.

Many window types and features are available. You can implement Horizontal menus and Multi-Input windows as part of your FOCUS application. Horizontal menus can also have Pulldown menus associated with each menu item.

You can perform a string search in an active window by entering any pattern followed by a blank and then pressing Enter. Within the pattern:

- An asterisk (*) is a multiple character wildcard.
- A question mark (?) is a single character wildcard.
- An equal sign (=) repeats the last string.

FOCUS tries to locate the line matching the pattern starting from the line following the current line. The search concludes at the line preceding the current line. If no match is found, a beep sounds and the cursor remains at the current position.
Introduction

The windows you can design with FOCUS Window Painter look just like the menus and screens you see in the FOCUS Talk Technologies, such as TableTalk and PlotTalk, but you can customize them to fit your application. You can design user-friendly menus and can display convenient and eye-catching instructions onscreen.

FOCUS Window Painter itself guides you step by step, using windows like those you will be creating.

On the windows you create, you can prompt users to:

- Select menu items from a list.
- Enter data.
- Select from automatically generated lists of available files and field names.
- Register a choice by pressing a function key.

You can also simply display explanations and instructions.

Window Painter is flexible enough to design the many different types of windows you might need for any application you can write with FOCUS.

You can also upload window files from FOCUS running in one operating environment, such as PC/FOCUS, and edit them using Window Painter for use on another operating environment such as MVS or CMS.

How Do Window Applications Work?

Window Painter stores the windows you design in window files. Window files work in conjunction with FOCEXEC procedures that use Dialogue Manager.

There are two major parts in any window application, each of which is a step for the developer:

- The windows, created with Window Painter, which users will see.
- The Dialogue Manager FOCEXEC.

You can invoke Window Painter to create and edit windows by typing

WINDOW [PAINT]

at the FOCUS prompt, and pressing Enter.

You can invoke the Window facility in your FOCEXEC by including the Dialogue Manager statement -WINDOW in the FOCEXEC. The -WINDOW statement provides the name of the window file, and the name of the individual window which should be displayed first. When the -WINDOW statement is executed by Dialogue Manager, control in the FOCEXEC passes to the Window facility.
The user is moved through the window file by goto values. A goto value tells the Window facility which window to display next.

You specify goto values when creating the windows with Window Painter. When your window is a menu with several items, you may assign a different goto value for each menu item, so that the next window depends on the user’s selection.

When you create the windows, you also specify return values. As with goto values, you may assign a different return value to each item on a menu. Return values are collected as the user moves through the windows, and are substituted for “amper variables” which can be used later in the window file or in the FOCEXEC when control passes back. (Amper variables are Dialogue Manager variables of the format &variablename.)

When the selected value is inserted in the FOCEXEC, you may test it with a Dialogue Manager IF...THEN statement and branch accordingly to a label in the FOCEXEC. In this way, you move the user through a series of windows, collecting return values for amper variables, using only one statement in your FOCEXEC.

You can use windows to collect amper variable values in place of any other method of prompting available through Dialogue Manager.

For a complete discussion of the Dialogue Manager facility, see Chapter 2, Managing Applications With Dialogue Manager. For details of integrating a FOCEXEC with the Window facility using return and goto values see Integrating Windows and the FOCEXEC on page 3-19.

Window Files and Windows

Windows—that is, menus and screens—are stored in window files. Windows are included in a specified window file as you create and save them during a Window Painter session.

- In CMS, window files have file type FMU, and are created and updated on the A disk automatically by Window Painter.
- In MVS, window files are contained in a partitioned data set (PDS) allocated to ddname FMU. Before any window files can be created, a PDS must be created and ddname FMU must be allocated to it.

Note, however, that creating a PDS is not necessary if you are creating window files to be used only in the current FOCUS session: Window Painter will temporarily allocate the PDS. For a full description of allocation requirements, see the appropriate Guide to Operations chapter in the FOCUS Overview and Operating Environments manual.

A window file can contain a maximum of 384 windows, and a number of windows may be displayed on the screen at once. All the windows in a single application may be stored together in one window file, or you may create separate window files for different parts of the application such as Help Windows.
Designing Windows With Window Painter

You can make an application more attractive by presenting menus in windows containing titles and other design elements, and can make an application easier to use by displaying function key definitions or other useful information.

Types of Windows You Can Create

Window Painter creates 10 different types of windows, each with its own special uses:

- Vertical Menus
- Horizontal Menus
- Text Input Windows
- Text Display Windows
- File Names Windows
- Field Names Windows
- File Contents Windows
- Return Value Display Windows
- Execution Windows
- Multi-Input Windows

These windows are described in the following sections.

The Vertical Menu

This is a Vertical Menu:

```
Select a report and press ENTER:

Accounts Payable
Accounts Receivable
Salary Information
Create a New Report
```

A menu is a window that lets users select an option from a list. These options are called menu items. A Vertical Menu lists its menu items one below the other. A user can select an item by moving the cursor down the list with the arrow keys and pressing Enter when the cursor is on the line of the desired item. A user can select more than one item if the window includes the Multi-Select option, which is part of the Window Options Menu. Help information can be specified for each item in the menu by using the menu-item help feature of help windows. For additional information on Multi-Select and Help windows see Window Options Menu on page 3-56.
The Horizontal Menu

This is a Horizontal menu:

```
+---------------------------------------------------+
| Reporting  Ad hoc     Maintenance   Quit         |
+---------------------------------------------------+
```

A Horizontal menu displays its menu items on a line, from left to right. You select an item by using PF11 or the Tab key to move right and PF10 or Shift+Tab to move left across the line, and pressing Enter when the cursor is at the desired item. You can also select an item by employing the search techniques available for FOCUS windows. (Search techniques are not available with pulldown windows).

If you use PF11 at the last item on the menu, the cursor moves to the first item on the menu. If you use PF10 at the first item on the menu, the cursor moves to the last item on the menu, unless there is another screen to scroll to.

An application can display an associated Pulldown menu for an item on a Horizontal menu when the cursor is on that item. Choose the pulldown option from the Window Options menu as discussed in Creating Windows on page 3-11. An option to display descriptive text above or below the Horizontal menu is also available from the Window Options menu.

You can assign any return value to each item on the menu. When you select a menu item, the corresponding return value is collected.

In a Horizontal or Vertical menu, you can assign a goto value to each menu item.

The Text Input Window

This is a Text Input window:

```
+-----------------------------------+
| Enter an Invoice Number:          |
+-----------------------------------+
```

Amper variables can be used in a Windows application. A Text Input window prompts the user to supply information needed in a FOCEXC. It is also possible to display an existing value to be edited. Each Text Input window accepts one line of input up to 76 characters long. You assign the length and format of the field when you create the window. Additional information about creating a Text Input window is found in Window Creation Menu on page 3-52.
The Text Display Window

This is a Text Display window:

```
+---------------------------------------------+
|Instructions for printing:|                      |
+---------------------------------------------+
|                                |
| Press ALT-7 if you wish         |
| to generate an OFFLINE report. |
|                                |
| Press ALT-8 if you wish         |
| to generate an ONLINE report.  |
+---------------------------------------------+
```

A Text Display window lets you present information such as instructions or messages. No selections can be made from a Text Display window, and no data can be entered in it.

The File Names Window

This is a File Names window:

```
+-------------------------------------------------------------+
|Select the report you wish to generate and press ENTER:|                  |
|+-------------------------------------------------------------+
|                   |
| SALARY FOCEXEC B1 |
| ACCTS FOCEXEC B1 |
| BILLS FOCEXEC B1 |
| BUDGET FOCEXEC B1 |
+-------------------------------------------------------------+
```

A File Names window presents a list of names of up to 409 files (in CMS) or 1023 PDS members (in MVS). The user can select one of these names by moving the cursor and pressing Enter when the cursor is on the line of the desired file name. You can specify selection criteria for the displayed file names when the window is created. A user can select more than one file if the window includes the Multi-Select option, which is available on the Window Options Menu.

Note that the maximum number of file (or member) names which can be displayed decreases as the width of the window increases. Narrower windows can display a greater number of names.
The Field Names Window

This is a Field Names window:

```
+---------------------------------------------------------------+
| Select the field you wish to sort on and press ENTER: |
+---------------------------------------------------------------+
| EMP_ID        | |
| LAST_NAME     | |
| FIRST_NAME    | |
| HIRE_DATE     | |
| DEPARTMENT    | |
| CURREN_SAL    | |
+(MORE)----------------------------------------------------------+
```

A Field Names window presents a list of all field names from a Master File; the user can select one by moving the cursor and pressing Enter when the cursor is on the line of the desired field name. A user can select more than one field if the window includes the Multi-Select option, which is available on the Window Options Menu.

You can use a Field Names window as the next step after a File Names window. That way, you can present a selection of files first, followed by the fields in a selected file.

The field names will be qualified when duplicates exist. You can use PF10 and PF11 to scroll left and right if a field name exceeds the maximum number of characters allowed on a line in a data field window.

Use PF6 as a three-way toggle to sort the fields in one of the following ways:

1. Display field names in the order in which they appear in the Master File.
2. Display field names in alphabetical order.
3. Display the fully qualified field names in the order in which they appear in the Master File.

The File Contents Window

This is a File Contents window:

```
+---------------------------------------------------------------+
| Select the record you want to display and press ENTER: |
+---------------------------------------------------------------+
| STAMFORD S 148 |
| NEW YORK U 142 |
| UNIONDALE R 77F |
| NEWARK U K1   |
+---------------------------------------------------------------+
```
The File Contents window displays the contents of a file. There is no limit on the size of a File Contents window. The user can select a line of contents by moving the cursor to it and pressing Enter. Each line can be up to 77 characters long. A user can select more than one line if the window includes the Multi-Select option, which is described as part of the Window Options Menu in Window Options Menu on page 3-56.

- In CMS, the contents of any file (except as noted below) can be displayed. You will be prompted for the file name and file type.

- In MVS, the contents of any member of a PDS (except as noted below) can be displayed. Sequential files can also be displayed in TSO. You will be prompted for a file name (the ddname) and a file type (the member name). This information should be entered as “member name ddname.”

Note: You cannot display a file with unprintable characters in a File Contents window. This includes files such as FOCUS files, HOLD files, SAVB files, FOCCOMP files, and encrypted files.

The Return Value Display Window

This is a Return Value Display window:

```
<table>
<thead>
<tr>
<th>This is a sample Return Value Display window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE FILE EMPLOYEE</td>
</tr>
<tr>
<td>PRINT EMP_ID FIRST_NAME LAST_NAME</td>
</tr>
<tr>
<td>END</td>
</tr>
</tbody>
</table>
```

The Return Value Display window displays amper variables that have been collected from other windows. No selections can be made from a Return Value Display window, and no data can be entered into it.

Return Value Display windows are very useful for constructing a command (or any string of words or terms) by working through a series of windows. An example of this type of application is seen when you construct a TABLE request using TableTalk.

Each line of the Return Value Display window is stored in a variable called &windownamexx, where windowname is the name of the window and xx is a line number.

Unless you use the Line-break option to place return values on separate lines, all collected return values are placed on the same line until the end of the line is reached. The length of the line is determined by the size of the window created. A description of the Line-break option on the Window Options Menu can be found in Window Options Menu on page 3-56.
Window Files and Windows

Only one Return Value Display window may be displayed at a time on the screen. It will collect a value from any active window (that is, a window from which a selection is being made or to which text is being entered, or an active Text Display window) if it is on that window’s display list. A description of the Display lists option on the Window Options Menu can be found in Window Options Menu on page 3-56.

You can clear the collected values from a Return Value Display window by including it on the hide list of a window which is being used. A description of the Hide lists option on the Window Options Menu can be found in Window Options Menu on page 3-56.

For a Multi-Select window, the Return Value Display window gives the number of selections, not the values selected. The values can be retrieved by using the -WINDOW command with the GETHOLD option.

The Execution Window

This is an Execution Window:

```
+-------------------------------------------------------------+
| → This is a sample Execution window.                          |
| TABLE FILE EMPLOYEE                                          |
| PRINT EMP_ID BY LAST_NAME                                    |
| END                                                           |
| →RUN                                                          |
+-------------------------------------------------------------+
```

The Execution window contains FOCUS commands such as Dialogue Manager statements, and TABLE requests.

You can create an Execution window by choosing its option on the Window Creation menu.

When this window is first displayed, it has a width of 77 characters, and no heading. You can place FOCUS commands within it. Note that the commands in an Execution window appear just as you type them; commands are not automatically converted to uppercase.
The Window Painter Main Menu contains an option enabling you to run a window in order to see any return values collected. If you were to run (not execute) the Execution window from the Window Painter Main Menu, you would see the Execution Window contents, then any windows called, and finally any return values collected by running the windows.

Note the following rules when using Execution windows:

- When you GOTO an Execution window, the contents of the window are executed. In all cases, execution begins at the top of the window.

- An Execution window is not displayed when executed, although the commands it contains may generate a display.

- An Execution window can use an amper variable as a goto value.

- An Execution window clears the screen and the Return Value display window.

- Execution windows have no return values.

- Execution windows can contain up to 22 lines.

- Execution windows can use local variables.

- Goto values for Execution windows should be assigned at line 1.

- Windows called from within Execution windows preempt window goto values. For example, a -WINDOW command issued from within an Execution window preempts an assigned goto value.

- The FOCUS statements within an Execution window follow normal Dialogue Manager execution (that is, FOCUS commands are stacked, Dialogue Manager commands are executed immediately). Any windows called from the Execution window will follow the logic determined by the windows themselves. This will substantially affect the application’s transfer of control.

- Use -RUN for immediate execution; otherwise requests will be performed after leaving the window application.

Normally, FOCUS returns to the window designated by the assigned goto value after the contents of the Execution window have been executed. However, when a jump is made to a window from inside an Execution window, the statements in the Execution window following the jump are skipped (along with any attached gotos). This differs from initiating a window from inside Dialogue Manager, which when finished returns you to the statement following the GOTO.
The Multi-Input Window

This is a Multi-Input window:

```
| Enter the following personnel information: |
|  Name:                                  |
|  Address:                                |
|    City:                                 |
|    Zip Code                             |
|  Phone Number:                          |
|  Department:                            |
```

A Multi-Input window prompts you for information that will be used in the application. A Multi-Input window may include up to 50 input fields, each of which can be up to 76 characters long. You assign the length, name, and format of the field when you create the window.

Use the Tab key to move the cursor between the fields on a Multi-Input window.

You can supply help information for each field in a Multi-Input window by using the Help window option. For information on Help windows, see Window Options Menu on page 3-56.

For a Multi-Input window, the return value is the name of the input field occupied by the cursor when you pressed Enter or a function key. The name that you supply for each input field is assigned to an amper variable with the same name as the field (each input field has a unique name). The variable &WINDOWVALUE contains the value of the input field occupied by the cursor when you pressed Enter or a function key.

Use a unique name for each field on a Multi-Input window. To display the field names specified, use the Input Fields option on the Window Options menu.

Creating Windows

The process of creating windows begins with choosing the type of window you want to create from the Window Creation menu. Each type of window requires slightly different instructions. The tutorial in Tutorial: A Menu-Driven Application on page 3-27 describes how to create and implement Text Display, Vertical menu, and File Names windows. This section describes how to create Horizontal menus (with or without associated Pulldown menus) and Multi-Input windows.
Creating a Horizontal Menu

To create a Horizontal menu, begin by placing the cursor at the Menu (horizontal) option on the Window Creation menu:

```
+-----------------------------------------------------+
| INSTRUCTIONS:  Move cursor to selection and hit ENTER |
| Use PF3 or PF12 to undo a selection                   |
| Use PF1 for help                                      |
+-----------------------------------------------------+
```

You will be prompted to enter a name and brief description for the window, after which you will reach the creation screen. On this screen:

1. Move the cursor to the location in which you want the top left corner of the menu to be displayed. Press Enter.

2. Next, use the arrow keys to move the cursor down (enough spaces to leave a line for each item you want to display as a menu choice) and to the right (enough spaces to just fit the longest menu item). Press PF4. You will see two windows: one is for entering information and the other is the corresponding Horizontal menu.

3. Enter the menu items in the window containing the cursor. Press the Enter key after each item; the item automatically appears on the Horizontal menu.
The following is an example of a completed creation screen:

```
+---------------------------------------------------------------+
| Vertical | Inputs | Lists | Execution | Misc | End |
+---------------------------------------------------------------+
```

Wind: HORD Typ: Menu (horz) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add

Once you have entered the items on your menu, there are several options you can select for each item. Move the cursor to any item and press PF2 to display the Window Options menu:

```
+----------------------+
| Exit this menu       |
| Goto value           |
| Return value         |
| FOEXEC name          |
| Heading              |
| Description          |
| Show a window        |
| Unshow a window      |
| Display list         |
| Hide list            |
| Popup (Off)          |
| Help window          |
| Link break           |
| Multi select (Off)   |
| Quit PF3             |
| Menu text            |
| Text line (x+1)      |
| Pulldown (Off)       |
| Conceal option       |
| Switch window        |
+----------------------+
```

Position the cursor on any option you want to select and press Enter.
Two features available for Horizontal menus are Menu text and Text line. Menu text is a line of text displayed when the cursor is on a menu item. The line on which the text is displayed is called the text line. You can position the text line one or two lines either above or below the Horizontal menu.

The following example illustrates Menu text and Text line. When the cursor is positioned on Vertical in the example below, the following is displayed:

```
+-----------------------------------------+
| Vertical | Inputs | Lists | Execution | Misc | End |
+-----------------------------------------+
```

In this example, the Menu text VERTICAL MENU TESTS is positioned at Text line x-1, one line above the menu. To place the Text line two lines above the Menu text, change x-1 to x-2. For Text lines below the menu text, use x+1 or x+2.

You can also select the Pulldown option for a Horizontal menu. With this option, you can assign a Pulldown menu to be displayed for a Horizontal menu item whenever the cursor is positioned on that item.

**Pulldown Menus**

When you set the Pulldown option ON, you can display an associated Pulldown menu for an item in a Horizontal menu by positioning the cursor on that item. The default is OFF; to change the setting to ON, position the cursor on the Pulldown option and press Enter. Note that when Pulldown is set ON, Menu Text is automatically set OFF.

The associated Pulldown menu must be a Vertical menu. When creating the Horizontal menu, you must assign a Goto value to point to the Pulldown menu. To do so, position the cursor on Goto value, press Enter, and enter the name of the Pulldown menu you want to display in the space provided:

```
| Reporting | rpts |
|-----------|
| Ad hoc    |
| Maintenance |
| Quit      |
```

You must create the Vertical menu, rpts, as you would any other Vertical menu. See *Tutorial: A Menu-Driven Application* on page 3-27 for examples.
The following example shows a Horizontal menu with the Reporting Pulldown menu displayed:

![Horizontal menu with Reporting Pulldown](image)

The following screen shows the same menu with the Ad hoc Pulldown menu displayed:

![Horizontal menu with Ad hoc Pulldown](image)

The following screen shows the same menu with the Maintenance Pulldown menu displayed:

![Horizontal menu with Maintenance Pulldown](image)

Note: To move from item to item in a Horizontal menu, use PF10 and PF11.
Creating a Multi-Input Window

To create a Multi-Input window, begin by placing the cursor at the Multi-Input window option on the Window Creation menu and press Enter. You will then be prompted for a name, description and heading. Place the window on the screen and size it as desired.

To place entries on the window:

1. Type the text for display.
2. Press PF6 at the point where the field begins.
3. Space along for the length of the field.
4. Press PF6 again to signify the end of the input area.
5. Enter name and information for the field.
The following example shows a Multi-Input window, with Name: entered as display text.

```
<table>
<thead>
<tr>
<th>Enter the following personnel information:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
</tr>
<tr>
<td>Address: XXXXXXXXXXXXXXXXXXXXXX</td>
</tr>
<tr>
<td>XXXXXXXXX , XX</td>
</tr>
<tr>
<td>Zip Code: XXXXXX - XXXX</td>
</tr>
<tr>
<td>Phone Number: XXX - XXX - XXXX</td>
</tr>
<tr>
<td>Department: XXXXXXXXXXX</td>
</tr>
</tbody>
</table>
```

This is what the developer’s screen looks like after several fields have been included in the Multi-Input window:

```
<table>
<thead>
<tr>
<th>Enter the following personnel information:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</td>
</tr>
<tr>
<td>Address: XXXXXXXXXXXXXXXXXXXXXX</td>
</tr>
<tr>
<td>XXXXXXXXX , XX</td>
</tr>
<tr>
<td>Zip Code: XXXXXX - XXXX</td>
</tr>
<tr>
<td>Phone Number: XXX - XXX - XXXX</td>
</tr>
<tr>
<td>Department: XXXXXXXXXXX</td>
</tr>
</tbody>
</table>
```

*Note:* Text fields may be supplied without headings or instructions. For example, see the city and state portion of the address line.
Designing Windows With Window Painter

This is how the window appears when run as part of the application:

```
+---------------------------------------------------------------------
| Enter the following personnel information:                           |
| +---------------------------------------------------------------------|
| | Name:                                                               |
| | Address:                                                            |
| | Zip Code                                                           |
| | Phone Number:                                                      |
| | Department:                                                        |
+---------------------------------------------------------------------
```

The following screen shows what is returned from the window when it is run inside the Window Painter:

```
+---------------------------------------------------------------------
| Variable     Value         |
| +---------------------------------------------------------------------|
| @WINDOWNAME   MULTI        |
| @WINDOWVALUE  |
| @MULTI       NAME          |
| @NAME        |
| @STREET      |
| @CITY        |
| @STATE       |
| @ZIP1        |
| @ZIP4        |
| @AREA        |
| @EXCHANGE    |
| @NUMBER      |
| @DEPARTMENT  |
| @ENTRY       |
| @ENTRYCODE   0             |
+---------------------------------------------------------------------
```

Note: To move from field to field in a Multi-Input window, use the Tab key.
The windows you create with Window Painter are designed to be used within an application FOCEXEC. This section discusses how to integrate your windows into your FOCEXEC.

**Syntax**

**The -WINDOW Statement**

To invoke the Window facility, insert the following Dialogue Manager statement in your FOCEXEC

```
-WINDOW windowfile winodname [PFKEY|NOPFKEY] [GETHOLD] [BLANK|NOBLANK] [CLEAR|NOCLEAR]
```

where:

- **windowfile**
  Identifies the file in which the windows are stored. In CMS, this is a file name. The file must have a file type of FMU or TRF. In MVS, this is a member name. The member must belong to a PDS allocated to ddname FMU.

- **windowname**
  Optional. Identifies which window in the file to display first. Can be set in Window Painter or in first window displayed.

- **PFKEY/NOPFKEY**
  Enables (prevents) testing for function key values during window execution.

- **GETHOLD**
  Retrieves stored amper variables collected from a Multi-Select window. Does not cause window to be displayed.

- **BLANK**
  Clears all previously set amper variable values when the -WINDOW statement is encountered. This is the default setting.

- **NOBLANK**
  No amper variable values are cleared when the -WINDOW statement is encountered.

- **CLEAR**
  When FOCUS is being used with the Terminal Operator Environment (described in the *Overview and Operating Environments* manual), the -WINDOW command clears the screen before displaying the first window. The Terminal Operator Environment screen will be redisplayed when control is transferred from the Window facility back to the FOCEXEC. This is the default setting.

- **NOCLEAR**
  When FOCUS is being used with the Terminal Operator Environment, the window file’s windows are displayed directly over the Terminal Operator Environment screens.

**Note:** NOBLANK is particularly important in applications which use more than one -WINDOW command.
Transferring Control in Window Applications

When the -WINDOW statement is encountered, control in the FOCEXEC is transferred to the Window facility. Control remains with the Window facility until one of the following occurs:

- The user makes a selection for which you have assigned no goto value.
- The PFKEY option is in effect and the user presses a function key (the function key must be set to RETURN, HX, CANCEL, or END, as described in the Testing Function Key Values on page 3-24.)

Once control passes back to the FOCEXEC, control only returns to the Window facility if another WINDOW statement is encountered.

Example Window File in an Application FOCEXEC

This example shows an application FOCEXEC and a window file named REPORT which contains three windows: R1, R2, and R3.

The numbers at the left of the example refer to the flow of execution (that is, the order in which the statements and windows are executed).

APPLICATION FOCEXEC

1. -START
2. -WINDOW REPORT R1 PFKEY
   -*
3. -*Control is transferred from the above statement
   -*to window R1 in window file REPORT.
   -*
4. -*IF &PFKEY EQ PF05 GOTO LABEL1;
   -*
   -*Control returns to the above statement from
   -*window R2 in window file REPORT.
   .
   .
   -LABEL1
5. -WINDOW REPORT R3
   -*
6. -*Control is transferred from the above statement
   -*to window R3 in window file REPORT.
   -*
7. -*IF &R3 EQ EXIT GOTO EXIT;
   -*
   -*Control returns to the above statement from
   -*WINDOW R3 in window file REPORT.
   .
   .
   -EXIT
Integrating Windows and the FOCEXEC

Note:

- At Step 3, the user selects an option from Window R1. This option’s goto value is R2. Control is transferred to Window R2.
- The user presses a function key in Window R2. Control is transferred to the FOCEXEC, to the statement following the -WINDOW command (Step 4).
- At Step 6, the user selects the option to exit; no goto value was set for that option. Control is transferred to the FOCEXEC, to the statement following the -WINDOW command (Step 7).

The flow of control has certain implications for the design of your window applications:

- Any time you wish to pass control back to the FOCEXEC, the window or menu option must have no goto value, or else must prompt the user to press a function key (as described in Testing Function Key Values on page 3-24).
- At some point in the window session, control should return to the FOCEXEC so that the accumulated return values can be substituted for amper variables, and the variables then used in the FOCEXEC.
- Any time you wish to pass control from the FOCEXEC to the Window facility you must insert the -WINDOW statement in the FOCEXEC.
- Note that it is not necessary to create a new window file for each -WINDOW command; you can simply enter the same file again at whatever window you wish.
- If you wish to test for a function key value in the middle of a series of windows, remember that pressing the function key automatically returns control to the FOCEXEC; an -IF test statement should follow the -WINDOW statement, and a second -WINDOW statement should be placed after the -IF statement to transfer control back to the window file.
- If you want to clear an existing set of variable values, you may do so by returning control to the FOCEXEC and executing another -WINDOW command with the BLANK option in effect.

To back up a step during window execution, the user may press the PF12 or PF24 keys. This will not cause control to pass to the FOCEXEC. However, you can force Dialogue Manager to return control to a FOCEXEC by a PF key setting as described in Testing Function Key Values on page 3-24.
Return Values

When the user responds to your window prompt by entering text, selecting an item from a menu, or pressing a function key, this response is the return value that fills in an amper variable in your FOCEXEC.

There are two ways in which amper variables are most commonly used in FOCEXECs:

- To collect values to plug into a FOCUS procedure such as a TABLE or GRAPH request so it can run.
- To test the value returned in a variable, and branch accordingly to a different part of the FOCEXEC or to another FOCEXEC.

The return value collected can be almost anything you desire: a character string, a number, the name of a file, a procedure name, or part of a FOCUS command.

A return value amper variable in the FOCEXEC has the same name as the window in which it is collected; that is:

&windowname

For example, the return value collected by the window MAIN supplies a value for the variable &MAIN.

- In Vertical and Horizontal Menu windows, you assign any return value you wish to each item on the menu. If the user selects that option, that return value is collected.
- In Text Input windows, the return value is the text that the user types.
- In Text Display windows, you can assign one return value to the entire window. Unlike other return values, a Text Display return value is collected as soon as control passes to the window, without the user needing to select anything.
- Return Value Display windows display return values collected from other types of windows. These return values can be displayed one per line, or several together on a single line. Although this type of window does not itself have a return value, each line has a corresponding amper variable (&windownamexx, where xx is the line number).
- For a Multi-Input Window, the return value is the name of the input field on which the cursor is positioned when you press Enter or a PF key.
- In windows with the Multi-Select option, the return value is the number of items selected.
- In File Names, Field Names, and File Contents windows, the return value is, respectively, the file name, field name, or line of file contents that the user selects from the display.
Example Return Value in a Menu-Driven Application

For example, assume that you have written a menu-driven application that enables a user to report from any one of a list of files. You have created a series of windows for this application, one of which is a File Names window named FILE designed to collect a return value for &FILE. The window displays a list of all the user’s files which meet certain file-identification criteria you specified when you created the window.

Your FOCEXEC contains these lines:

-START
-WINDOW EXAMPLE FILE
  .
  .
  TABLE FILE &FILE

When the user moves the cursor to SALES and presses ENTER, SALES is collected to be substituted for &FILE in the FOCEXEC:

TABLE FILE SALES

Goto Values

When you are creating your windows, you will also assign goto values telling the Window facility which window to display next. These values allow you to move the user through a series of windows, collecting return values for amper variables, without adding lines to your FOCEXEC.

• In Vertical and Horizontal Menu windows, you assign a goto value for each menu item.
• In all other windows, you assign a single goto value.
• You can use an amper variable as a GOTO value.

As described in Transferring Control in Window Applications on page 3-20, if you assign no goto value to a menu option or window, control passes back to the FOCEXEC when the user selects that option or presses Enter at that window.

It is important not to confuse these goto values with the Dialogue Manager -GOTO statement. The goto value points your application to a new window in the window file; the -GOTO statement transfers control to a label in your FOCEXEC.

Returning From a Window to Its Caller

You can return from a window to its caller via the <ESCAPE> option. If you enter this string as the goto value of a window, control will return to the previous window upon completion of the current window (enter the right and left carets as part of the goto value).
Window System Variables

We have already discussed return values: these are specific to each window. Two other Window facility variables, &WINDOWNAME and &WINDOWVALUE, are specific to the -WINDOW session (not to each window) and receive their values when the Window facility passes control from a window file back to the FOCEXEC.

&WINDOWNAME

&WINDOWNAME is an amper variable containing the name of the last window that was displayed before the Window facility transferred control back to the FOCEXEC.

This variable can be used in many ways. For example, if the goto values/function key prompts in a window file allow a user to leave the window file from several different windows, you can test &WINDOWNAME in the FOCEXEC to determine which window the user was in last (and, therefore, which path the user navigated through the window file).

&WINDOWVALUE

&WINDOWVALUE is an amper variable containing the return value from the last window that was displayed before the Window facility transferred control back to the FOCEXEC. If the user selected a line for which no return value was set (for example, a blank line between two menu options in a Vertical Menu window), then &WINDOWVALUE will contain the line number of the line that was selected.

This variable can be used in many ways. For example, if the goto values/function key prompts allow a user to leave the window file from several different windows, and you need to know the return value of the last window the user was in before she or he left the file by pressing a function key, you can test &WINDOWVALUE.

Testing Function Key Values

If you wish to test for function key values, you must specify the PFKEY option on the -WINDOW command line. When the PFKEY option is set and a user presses a function key during window execution, the name of that key is stored in the amper variable &PFKEY.

For example, if the user presses PF1, the 4-character value of &PFKEY is PF01; if PF2, the value is PF02, and so forth. If the user presses Enter, the value is ENTR. The value of &PFKEY is reset each time the user presses a function key.

Note that if the PFKEY option is specified, the Window facility’s default PF key actions are overridden by the general FOCUS PF key settings. This means that when you specify the PFKEY option, if you still want the standard Window facility PF key actions to be available to window users (for example, PF1 = HELP, PF3 = UNDO), you must use the SET command in your application FOCEXEC, followed by a -RUN statement, to explicitly set those actions.
Integrating Windows and the FOCEXEC

For example, if you specify the PFKEY option but you want to retain all of the Window facility’s default PF key actions using the same PF keys, you need to include the following commands before the -WINDOW statement in your application FOCEXEC:

```
SET PF01=HELP
SET PF03=UNDO
SET PF04=TOP
SET PF05=BOTTOM
SET PF06=SORT
SET PF07=BACKWARD
SET PF08=FORWARD
SET PF09=SELECT
SET PF10=LEFT
SET PF11=RIGHT
SET PF12=UNDO
```

When you specify the PFKEY option, any PF key which you want to test for in the application FOCEXEC must be set to RETURN. (HX, CANCEL, and END also function as RETURN within the Window facility, and can be used in place of it.)

For example, if you design your application so that a user can press PF2 to choose an additional menu option, and therefore you want to test &PFKEY for the value PF02 in your application FOCEXEC, then you must include the following SET command before the -WINDOW command in your application FOCEXEC:

```
SET PF02=RETURN
```

The SET PF command is discussed in Chapter 4, Setting Parameters: SET, and in the Maintaining Databases manual.

You can list the current general FOCUS PF key settings by issuing the ? PFKEY command. The ? PFKEY command is discussed in Chapter 6, FOCUS Query Commands.

The variable &PFKEY can be tested just like any other amper variable. Note that the name of the variable is always &PFKEY; it is not linked to a window name like other amper variables collected through windows.

You may test the PFKEY variable repeatedly throughout the FOCEXEC. Additional SET statements are not required.

One of the advantages of using the &PFKEY variable is that it enables you to collect two return values from a single menu. You might, for example, create a window called FILES, which prompts the user to enter the name of a file, then press PF7 to produce a graph or PF8 to produce a report. Both the file name as &FILES and the function key value as &PFKEY would be collected as return values.

It is always important to remember that pressing a function key will immediately return control to the FOCEXEC if that key was set to RETURN (or to HX, CANCEL, or END).
Designing Windows With Window Painter

**Note:** If the cursor is on a menu that has a FOCEXEC associated with it, the FOCEXEC is executed and the GOTO value associated with the menu choice is assumed. The PFKEY is ignored.

In the example above, if the user presses a function key before typing the file name, the &FILES variable will not be collected. If the key was set to something other than RETURN, HX, CANCEL, or END, then the action it was set to is invoked, and control remains within the Window facility.

### Executing a Window From the FOCUS Prompt

You can execute a window directly from the FOCUS command prompt.

**Syntax**

**How to Execute a Window From the FOCUS Prompt**

The syntax is:

```plaintext
EX 'windowfile FMU' [windowname] [PFKEY|NOPFKEY] [BLANK|NOBLANK] [CLEAR|NOCLEAR]
```

where:

- **windowfile**
  - Is the file containing the windows. It must have file type FMU, and appear within single quotation marks.

- **windowname**
  - Identifies the first window to be executed. If a window name is not specified, FOCUS will execute the default start window, or the first window created.

- **PFKEY/NOPFKEY**
  - Tells FOCUS you will (will not) be testing for function key values during execution.

- **BLANK**
  - Clears previously set amper variables when the window is called. This is the default setting.

- **NOBLANK**
  - Retains previously set amper variables.

- **CLEAR**
  - When FOCUS is being used with the Terminal Operator Environment, the screen is cleared when the EX command is encountered. The Terminal Operator Environment screen is restored when the last window in the chain has been executed. This is the default setting.

- **NOCLEAR**
  - When FOCUS is being used with the Terminal Operator Environment, the screen is not cleared when the EX command is encountered, and any windows are displayed within the Terminal Operator Environment screens.
Tutorial: A Menu-Driven Application

For example, to execute the window MAIN in the window file REPORT, you could issue EX ‘REPORT FMU’ MAIN from the FOCUS command prompt, which is equivalent to issuing -WINDOW REPORT MAIN from Dialogue Manager.

Tutorial: A Menu-Driven Application

This tutorial describes a menu-driven system that clerical personnel can use to produce sales reports and graphs at your chain of retail stores. The system must fulfill three major requirements:

- **Ease of use.** Your system must let employees be productive without extensive training.
- **Functionality.** The system has to work properly with only a few steps.
- **Appearance.** There should be continuity between screens, and a general unity of design. The reports and graphs produced must be attractive and easy to read.

The application prompts the user to select reporting or creating a graph.

Then, the user may opt to execute an existing FOCUS request or to create a new one. A user who chooses to execute an existing request will be shown an automatically generated list of FOCEXECs from which to pick. A user who chooses to create a new request will be placed in either TableTalk or PlotTalk, depending on whether reporting or creating a graph was chosen in the first step.

While the report or graph is being generated, a corresponding message will be displayed on the terminal screen. And, after the output is displayed, the user can choose to generate another report or graph, or else to exit.

The following figure illustrates the logic of the application.

APPLICATION FOCEXEC

-START
-WINDOW SAMPLE MAIN
-"Control is transferred from the above statement
to window MAIN in window file SAMPLE.
-"-IF &MAIN ...
-"Control returns to the above statement
-"from option ”Exit?” in window MAIN,
-"from option ”New Request?” in window EXECTYPE,
-"and from every selection in window EXECNAME.
-"
-"
-"GOTO START
-EXIT
Creating the Application FOCEXEC

A FOCEXEC called SAMPLE will drive this application.

Begin by using the TED editor to create the FOCEXEC file SAMPLE. At the FOCUS prompt, type

TED SAMPLE

and press Enter. (In CMS, TED assigns the file type FOCEXEC unless you specify another file type. In MVS, you must specify ddname as follows:

FOCEXEC (SAMPLE)

Type in the following FOCEXEC. Note that the numbers on the left refer to explanatory notes. Do not type them in your FOCEXEC file, but read the notes as you go along. All commands that begin with a hyphen, such as -WINDOW, are Dialogue Manager statements, and they must begin in the first column. Dialogue Manager is discussed in Chapter 2, Managing Applications With Dialogue Manager.

<table>
<thead>
<tr>
<th>Window</th>
<th>If option selected is:</th>
<th>Then go to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>Report?</td>
<td>window EXECTYPE</td>
</tr>
<tr>
<td></td>
<td>Graph?</td>
<td>window EXECTYPE</td>
</tr>
<tr>
<td></td>
<td>Exit?</td>
<td>back to FOCEXEC</td>
</tr>
<tr>
<td>EXECTYPE</td>
<td>Existing Request?</td>
<td>window EXECNAME</td>
</tr>
<tr>
<td></td>
<td>New Request?</td>
<td>back to FOCEXEC</td>
</tr>
<tr>
<td>EXECNAME</td>
<td>The options in this window are a list of report and graph requests from which the user can select.</td>
<td>Control is transferred back to the FOCEXEC</td>
</tr>
</tbody>
</table>
You will notice that this application determines variable values in two ways: there are variables for which values are collected by windows, and variables which are set within the FOCEXEC using the -SET statement.

-START
1. -WINDOW SAMPLE MAIN
2. -IF &MAIN EQ XXIT GOTO EXIT;
   -IF &MAIN EQ RPT GOTO GENERATE;
   -IF &MAIN EQ GRPH GOTO GENERATE;
   -GOTO START
-************************** GENERATE **************************
3. -GENERATE
4. -IF &EXECTYPE EQ EXIST GOTO RPTEX ELSE GOTO NEWRPT;
5. -RPTEX
6. EX &EXECNAME
7. -SET &FORMAT=IF &MAIN EQ RPT THEN REPORT
   -ELSE IF &MAIN EQ GRPH THEN GRAPH;
8. -TYPE GENERATING &FORMAT
9. -RUN
10. -GOTO START
11. -NEWRPT
12. -SET &PROCNAME=IF &MAIN EQ RPT THEN TABLETALK
    -ELSE IF &MAIN EQ GRPH THEN PLOTTALK;
13. &PROCNAME
14. -RUN
15. -GOTO START
-****************************** EXIT ******************************
16. -EXIT

1. The -WINDOW statement transfers control to the Window facility. SAMPLE is the name of the window file this application will use. (We will create it in this tutorial.) MAIN is the window where the procedure will begin.

   Control will not return to the next line of the FOCEXEC until a window is processed for which no goto value has been assigned, in this case, EXECTYPE or EXECNAME.

2. The return value collected for &MAIN—collected from the window MAIN—is tested. The FOCEXEC branches to a label depending on its value.

   If the return value for &MAIN is RPT or GRPH, the FOCEXEC will branch to -GENERATE; if XXIT, to -EXIT. Each return value corresponds to a selection on the menu window MAIN.

3. This label begins the GENERATE section of the FOCEXEC.
Designing Windows With Window Painter

4. The value collected for &EXECTYPE (from window EXECTYPE) is tested and the FOCEXEC branches accordingly. Note that this value was collected from the window EXECTYPE while the Window facility was in control, without a prompt from Dialogue Manager.

5. This label begins the RPTEX section of the FOCEXEC.

6. The FOCUS command which will execute an existing report is stacked. The value of &EXECNAME—the name of the existing report—was collected while the window file was in control. The single quotation marks around &EXECNAME tell FOCUS to treat the value—which may contain more than one word (in CMS, for example, a file name and a file type)—as part of a single file identification.

7. The value of the variable &FORMAT is set according to the return value from the MAIN window. If the value was RPT, &FORMAT is set to REPORT; if the value is GRPH, &FORMAT is set to GRAPH.

8. A message containing the value of &FORMAT is displayed for the user while the stacked FOCUS request is executing.

9. -RUN executes the stacked command(s).

10. When the request output has been displayed, the FOCEXEC branches back to -START, where the user can choose to exit or to create another report or graph. All amper variable values collected in the previous round are cleared when the -WINDOW command is encountered.

11. This label begins the section NEWRPT.

12. This statement sets the value of &PROCNAME to TABLETALK if the value of &MAIN is RPT, to PLOTTALK if the value is GRPH.

13. This line stacks the command TABLETALK or PLOTTALK.

14. -RUN executes the stacked command.

15. This statement returns to -START, as in note 10.

16. This statement ends FOCEXEC execution.
Creating the Window File

The -WINDOW statement SAMPLE FOCEXEC tells FOCUS to look for a window file named SAMPLE and a window named MAIN. The complete list of windows used in this application is:

<table>
<thead>
<tr>
<th>Window Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORDER</td>
<td>A Text Display window used as a background display for the other windows.</td>
</tr>
<tr>
<td>BANNER</td>
<td>A Text Display window that introduces the application.</td>
</tr>
<tr>
<td>MAIN</td>
<td>A Vertical Menu from which the user can choose to create a graph or a report, or exit the application.</td>
</tr>
<tr>
<td>EXECTYPE</td>
<td>A Vertical Menu from which the user chooses to execute an existing procedure or create a new one.</td>
</tr>
<tr>
<td>EXECNAME</td>
<td>A File Names window displaying all FOCEXEC files, from which the user can select one to execute. This window is seen only if the user opts to execute an existing report in EXECTYPE.</td>
</tr>
</tbody>
</table>

All these windows will be included in the window file named SAMPLE. You are going to start by building that window file.

- In CMS, when you use Window Painter to create a window file, the file is automatically created by the system on your A disk.
- In MVS, before you can use Window Painter to create a window file, a PDS must be allocated with ddname FMU, LRECL 4096, and RECFM F. BLKSIZE 4096 is recommended.

You can reach the FOCUS Window Painter Entry Menu by typing

```text
WINDOW [PAINT]
```

at the FOCUS prompt, and pressing Enter.

The Entry Menu is the first screen you see:

```
FOCUS WINDOW PAINTER

INSTRUCTIONS: Move cursor to selection and hit ENTER
Use PF3 or PF12 to undo a selection
Use PF1 for help

Select the window file:

New File Create a new file
TEST This is a test.
```
Designing Windows With Window Painter

Since you are creating a new window file, choose NEW FILE, and press Enter. The next screen you see prompts you to name the window file.

Since the FOCEXEC will look for a window file named SAMPLE, type

SAMPLE

and press Enter.

```
FOCUS WINDOW PAINTER
+-----------------------------------------------+
| INSTRUCTIONS : Move cursor to selection and hit ENTER |
| Use PF3 or PF12 to undo a selection              |
| Use PF1 for help                                |
+-----------------------------------------------+

Enter the window file name: |

SAMPLE
```

You will see a screen asking for a description of the window file.

Type

Sample file for Window Painter tutorial

and press Enter.

```
FOCUS WINDOW PAINTER
+-----------------------------------------------+
| INSTRUCTIONS : Move cursor to selection and hit ENTER |
| Use PF3 or PF12 to undo a selection              |
| Use PF1 for help                                |
+-----------------------------------------------+

Enter a description: |

Sample file for Window Painter tutorial.
```
Creating the Text Display Window Named BORDER

Now you are ready to create the first window. The screen that appears on your display is the Window Painter Main Menu. Select

Create a new window

and press Enter.

The Window Creation Menu asks what kind of window you want to create.

[Menu options...]

The Window Creation Menu asks what kind of window you want to create.
Designing Windows With Window Painter

The BORDER window is the first window you will create for the application. BORDER will supply a background border for other windows. It is a Text Display window, so select Text display and press Enter.

Next, you are asked to name the window. Type BORDER and press Enter.

The Window Description Screen appears next. This description does not appear when the window is displayed, but becomes part of the document file that Window Painter creates describing all windows in the file. Since the document file is very useful when writing your FOCEXEC, it is a good idea to enter a functional description here. To describe this window, type

This window borders all my screens.

and press Enter. The ability to annotate screens in this manner is very useful when selecting windows to edit.

The Window Heading Screen comes next. Since you do not want a heading displayed on this window, simply press Enter to bypass it.
The Window Design Screen displayed now is nearly blank, with a cursor for you to position where you want the upper left-hand corner of BORDER to be. Leave the cursor where it is and press Enter.

A small box appears around the cursor; this is the window. You will now make the window larger. Using the arrow keys, move the cursor to the right edge of the screen, on the line just above the status line: this will be the new lower right corner of the window. Now press PF4 to resize the window. (PF4 functions as the SIZE key in the Window Design Screen.) The window has been resized so that its lower right corner is where you positioned the cursor: the window now fills the entire screen.

When resizing a window, remember that the window’s lower right corner refers to the lower right corner of the window border, which is shown as a plus sign (+) on the screen. It is this corner that you are moving when you resize the window. On the other hand, the last row of the window refers to the last row which can contain data or text: this is the row immediately above the bottom border.

This window’s border will form the background border for the other windows in this application.

If you need help using the keyboard while in the Window Design Screen, press PF1 (the Window Painter Help key) to see the following display:

```
File: SAMPLE FOCUS WINDOW PAINTER

Help: Text display and Return value display windows
Use the arrow keys to move the cursor around on the screen.
To enter text for a line, simply type that text in the window,
for text display.

PF01: Help
PF02: Main options menu
PF03: Quit the Menu Design Screen.
PF04: Resize the window.
If you find that you do not have enough room in the window to
type the text you want, move the cursor to where you want the
new lower-right-hand corner to be, and press PF04 or PF16.
PF05: Set a window to go to if the current line is selected.
PF06: Set a return value for the current line.
PF07: Move the window.
To move the window, place the cursor where you want the new
upper-left-hand corner to be, and press PF09 or PF21.
PF10: Delete the line that the cursor is on.
PF11: Insert a line at the cursor position.
```
Now that the window is complete, you should save it. Press PF3.

Press Enter to select Save. You will be returned to the Main Menu.

Creating the Text Display Window Named BANNER

BANNER is also a Text Display window, but is smaller than BORDER and contains text that identifies this application.

From the Window Painter Main Menu, select

Create a new window

and press Enter. Select

Text Display

and press Enter. The name of this window is

BANNER

and its description is:

Banner for application MAIN menu.

Enter this name and description just as you did for the BORDER window. When prompted for a heading, press Enter.
At the Window Design Screen, use the arrow keys to move the cursor two spaces to the right, and press Enter. Now position the cursor 64 more spaces to the right and two rows down, and press PF4 to resize the window.

You will now enter text to be displayed in the window. Reposition the cursor on the first line within the window, ten spaces to the right of the window’s left border, and type:

The Milkmore Farms Weekly Reporting System

Type a line of asterisks (*) all the way across the window’s second line. (Begin at the second column within the window, because the first column of every window is protected.)

You will now center the banner in the width of the screen. Estimate where the upper left corner of the window would be if the window were centered. Position the cursor there, and then press PF9. The window moves to its new location. Repeat the process if you need to center it more precisely.

The window should look like this:

```
+------------------------------------------------------------------------+
| The Milkmore Farms Weekly Reporting System                             |
| +------------------------------------------------------------------------+
```

Press PF3 and save the window.
Creating the Vertical Menu Window Named MAIN

Now you will create the MAIN vertical menu window, which collects the amper variable \&MAIN. Select

Create a new window

and press Enter.

BORDER and BANNER are Text Display windows, from which no options may be selected. Since MAIN, however, is a menu from which a selection must be made, choose

Menu (vertical)

and press Enter. Name the window:

MAIN

On the Description screen, type

User can report, graph, or exit.

and press Enter.

When prompted for a heading, type ten spaces, then

Would you like to:

and press Enter.

On the Window Design Screen, move the cursor five rows from the top and 20 columns from the left, and press Enter. The window will be created wide enough to contain the heading. Now position the cursor six rows below the window’s bottom edge, and ten columns to the right of its right edge. Press PF4 and the window will be resized.

Type the following menu options as they appear below:

```
*--------------------------------------------------*
| Would you like to:                |
|--------------------------------------------------*
| Create a report?                  |
| |                                  |
| Create a graph?                   |
| |                                  |
| Exit?                              |
| |                                  |
*--------------------------------------------------*
```

Wind: MAIN Type: Menu (vert) PF1=Help 2=Menu 4=Size 9=Move 10=Del 11=Add
Now you will assign goto and return values for each menu option. To assign either value to an option, the cursor must first be on that option.

Move your cursor back to

Create a report?

and press PF2 to display the pop-up Window Options Menu.

Assigning a goto value tells the Window facility to display another window when this item is selected during execution.

In the next window of this application, the user will be prompted to either execute an existing report or create a new one. The window which displays that prompt will be called EXECTYPE, so the goto value of the first two menu options will be EXECTYPE.

Move the cursor to

Goto value

and press Enter.
In the space provided, type

```
!EXECTYPE
```

and press Enter.

The return value collected by this window—&MAIN—will be tested in the FOCEXEC:

```
-START
-WINDOW SAMPLE MAIN
-IF &MAIN EQ XXIT GOTOEXIT;
-IF &MAIN EQ RPT GOTO GENERATE;
-IF &MAIN EQ GRPH GOTO GENERATE;
.
.
.
```

Now move the cursor to

Return value

and press Enter.
Type the value
RPT
as shown, and press Enter.

Exit the Window Options Menu by moving the cursor to
Exit this menu
and pressing Enter.

Now you will set the values for:
Create a graph?
Move the cursor to the second menu item, and press PF2.
Repeat the steps you just performed, assigning the goto value
EXECTYPE
and the return value:
GRAPH
Leave the Window Options menu and move the cursor to
EXIT?
For this option, you will not assign a goto value. Since it exits to the FOCEXEC, there is no
next window to be displayed.
Repeat the steps to assign the return value:
XXIT
With the Window Options Menu still on the screen, move the cursor to
Display list
and press Enter.

The display list may specify up to 16 windows to be displayed when this window is visible
during execution. Since you want BORDER and BANNER to be displayed with MAIN, you
must add them to the list.

Select:
Add to the list

A list of windows appears, from which you select by moving the cursor and pressing Enter. The
windows must be selected in the order in which they should appear, because they will be
overlaid one on top of another when displayed. Select BORDER and BANNER for MAIN’s
display list, being certain to select BORDER first so that it will be displayed behind BANNER.

When you have finished, choose Quit to return to the Window Options Menu.

Quit the Window Options Menu and press PF3 to save MAIN.

Before moving on, look at what you have done so far. Select
Run the window file
and press Enter.
Select

MAIN

as the starting screen. Press Enter, and you will see a screen like this:

```
The Milkmore Farms Weekly Reporting System

Would you like to:

Create a report?

Create a graph?

Exit?
```

Position the cursor on the “Create a report” line. When you press Enter to continue the display, you will see an error message because EXECTYPE—the goto value—has not been created yet. Ignore it, and press Enter to continue. You will see a screen displaying amper variables for this window and their values. Press Enter to return to the Main Menu.

**Creating the Vertical Menu Window Named EXECTYPE**

So far you have created two Text Display windows and a Vertical Menu. The next window we will create will also be a Vertical Menu.

Select

Create a new window

from the Main Menu, and choose

Menu (vertical)

from the Window Creation Menu. Enter

EXECTYPE

as the window name.
When prompted for a description, type

Create a new FOEXEC or run existing one

and press Enter. When prompted for a heading, press Enter.

When the Window Design Screen appears, move the cursor 12 rows down the screen and 22 columns to the right, and press Enter. Now reposition the cursor four rows beneath the bottom edge of the window and 32 columns to the right of the right edge of the window, and press PF4 to resize it.

Type the following two menu options as they appear below:

When you created the MAIN window, you used the Window Options Menu to set each return value and goto value. There is an easier way to set return and goto values using the PF6 and PF5 keys.

Pressing PF5 prompts you successively for a GOTO value, a Return value and a FOEXEC name. When prompted for a GOTO value press Enter again and you will be prompted for the Return value. Enter EXIST and press PF5 again and you are prompted for FOEXEC name. Just press Enter.

If you select

... using an existing request.

from the EXECTYPE menu, the File Names window EXECNAME will be displayed next. EXECNAME will contain a list of existing FOEXEC files from which you may choose.

Move the cursor to the second menu item.

Now you need to consider the return and goto values for this option.
If you choose to create a new report or graph request, EXECNAME will not be displayed. Rather, control must pass back to the FOCEXEC, which will execute these lines:

```
-IF &EXECTYPE EQ EXIST GOTO RPTEX ELSE GOTO NEWRPT;
-NEWRPT
-SET &PROCNAME=IF &MAIN EQ RPT THEN TABLETALK ELSE IF &MAIN EQ GRPH THEN PLOTTALK;
&PROCNAME
-RUN
```

Since you want control to pass to the FOCEXEC if this option is chosen, you will not assign a goto value to it. Remember that during execution control passes to the FOCEXEC when an option without a goto value is selected.

The return value may be anything other than EXIST. For now, press PF6, and enter NEXIST.

Rather than create display and hide lists for EXECTYPE, make it a pop-up window. A pop-up window is displayed like any other window, but disappears when the user presses Enter. EXECTYPE pops up in front of MAIN.

Press PF2 to display the Window Options Menu, move the cursor to Popup(Off) and press Enter. You will see that (Off) changes to (On).

Exit the Window Options Menu, press PF3, and save the window.

**Creating the File Names Window Named EXECNAME**

Your final window is the File Names window which displays a list of existing FOCUS report requests. On the Window Creation Menu, select:

- File names
- Name the window EXECNAME
- and type in the description:
  Select an existing FOCEXEC from list.
- Enter an explanatory heading:
  Select the request you want to execute and press ENTER:
Designing Windows With Window Painter

You will be prompted for file-identification criteria. Type

* FOCEXEC

and press Enter.

In CMS, when the application is executed, this will select all files having the file type FOCEXEC.

In MVS, when the application is executed, this will select all members of ddname FOCEXEC.

On the Window Design Screen, move the cursor two rows down and press Enter. Use PF9 to center the window on the screen. Resize the window: reposition the cursor two columns to the right of the window’s right edge and ten rows below the window’s bottom edge, and press PF4.

Since only BORDER should be displayed with this window, add BANNER, MAIN, and EXECTYPE to the hide list and add BORDER to the display list.

When the user selects a file name from this window during execution, that file name will automatically be collected as the return value. You cannot set the return value any other way for this type of window.

In the FOCEXEC, that return value will be plugged into the line

EX &EXECNAME

and the report or graph request will be executed.

But in order for this to happen, you must return control to the FOCEXEC. Therefore, you will assign no goto value to this window.

If you want to change the file identification criteria of a File Names window (or of a Field Names or File Contents window) after it has been created, change the “return value.” Although these two window types cannot have their actual return values set when the window is created or edited, the “return value” which is displayed and can be set is actually the window’s file identification criteria. You can change the file identification criteria just as you would change the actual return value of a Vertical Menu window.

Exit from the Window Options Menu, press PF3, and save the window. The window file is complete. Exit from Window Painter.
**Executing the Application**

To execute the SAMPLE FOCEXEC, at the FOCUS prompt, type

```
EX SAMPLE
```

and press Enter. When prompted to choose a new or existing FOCEXEC, select

```
... using a new request.
```

unless you have created one in an earlier FOCUS session. The application will execute PlotTalk or TableTalk. If you save the request you create, you can try the SAMPLE FOCEXEC again, and execute the new request by selecting:

```
... using an existing request.
```

This completes the tutorial.

**Window Painter Screens**

The creation of windows is itself an automated window-driven process. There are six major screens:

- The Entry Menu
- The Main Menu
- The Window Creation Menu
- The Window Design Screen
- The Window Options Menu
- The Utilities Menu

These screens assist you whenever you create or edit windows.
Invoking Window Painter

To invoke Window Painter, type the WINDOW PAINT command at the FOCUS prompt and press Enter.

**Syntax**

**How to Invoke Window Painter**

The syntax of the command is

```
WINDOW [PAINT [filename]]
```

where:

- **PAINT**
  - Is optional.

- **filename**
  - Is the name of the window file that you want to work with.

In CMS, this is a file name. The file must have a file type of FMU.

In MVS, this is a member name. The member must belong to ddname FMU.

If you do not specify file name, you will begin your Window Painter session at the Entry Menu, where you can choose a window file to use or can create a new window file. If you do specify file name, you will skip the Entry Menu and begin your Window Painter session at the Main Menu, working with the window file you specified.

If the file name does not exist, you will be asked if you want to create a new file. If not, the Window Painter Entry Menu will be displayed.
Entry Menu

You can reach the Window Painter Entry Menu by typing

```
WINDOW [PAINT]
```

at the FOCUS prompt, and then pressing Enter.

The Entry Menu is the first screen you see:

```
File: SAMPLE  FOCUS  WINDOW  PAINTER

+-----------------------------------------------+
|   INSTRUCTIONS : Move cursor to selection and hit EMIER |
|   Use PF3 or PF12 to undo a selection              |
|   Use PF1 for help                                |
+-----------------------------------------------+

+-----------------------------------------------+
| Select the window file:                       |
+-----------------------------------------------+
| New File  Create a new file                   |
| TEST     This is a test.                      |
| SAMPLE   Sample file for Window Painter tutorial. |
+-----------------------------------------------+
```

The Entry Menu invites you to choose a window file in which to work. If you are creating windows for a new application, you should start a new window file. If you are maintaining or creating windows for an existing application, use the window file that corresponds to your application.

When you become comfortable working with windows, you can write FOCEXECs that include branching between window files. Refer to *Transferring Control in Window Applications* on page 3-20 for a detailed discussion on branching and transferring control.
Main Menu

Once you have selected a window file from the Entry Menu, or entered the WINDOW PAINT command with the file name option, the Main Menu appears:

```
FOCUS WINDOW PAINTER

*---------------------------------------------------*
| INSTRUCTIONS : Move cursor to selection and hit ENTER |
| Use F1 or F12 to undo a selection                   |
| Use F1 for help                                    |
*---------------------------------------------------*

Select one of the following:

- Create a new window
- Edit an existing window
- Delete an existing window
- Run the window file
- Switch window files
- Utilities
- End
- Quit without saving changes

*---------------------------------------------------*
```

The following table summarizes the options on the Main Menu, along with illustrations of screens that appear when you select some of the options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new window</td>
<td>Brings up the Window Creation Menu. You can select the type of window you want to create.</td>
</tr>
<tr>
<td>Edit an existing window</td>
<td>Brings up a list of windows in your current window file. You can select the one you want to edit.</td>
</tr>
</tbody>
</table>
Delete an existing window

Brings up a list of windows in your current window file. You can select the one you want to delete.
Designing Windows With Window Painter

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the window file</td>
<td>Brings up a list of windows in your current window file. You can select the one from which you want to start running the window file. After the window file is run, the windows’ amper variable values are displayed. The display includes the first 20 characters of each value. This option shows you how your windows work without executing the FOCEXEC. Use this option to test your window file.</td>
</tr>
<tr>
<td>Switch Window files</td>
<td>Returns you to the Window Painter Entry Menu, from which you can select another window file. The previous window file is saved whenever you switch window files.</td>
</tr>
<tr>
<td>Utilities</td>
<td>Brings up the Utilities Menu, which is discussed in <em>Utilities Menu</em> on page 3-67.</td>
</tr>
<tr>
<td>End</td>
<td>Returns you to native FOCUS. All work which you saved during the Window Painter session is kept.</td>
</tr>
<tr>
<td>Quit without saving</td>
<td>Returns you to native FOCUS. All work which you saved during the Window Painter session is discarded.</td>
</tr>
</tbody>
</table>

Window Creation Menu

You can reach the Window Creation Menu by selecting Create a New Window from the Main Menu. You will see the following screen

```
+-------------------------------------------------------------+
| INSTRUCTIONS :  Move cursor to selection and hit ENTER      |
|                Use PF3 or PF12 to undo a selection              |
|                Use PF1 for help                               |
+-------------------------------------------------------------+

|---------------------------+|
| Select the window type:   |
|---------------------------+|
| Menu (vertical)           |
| Menu (horizontal)         |
| Text input                |
| Text display              |
| File names                |
| Field names               |
| File contents             |
| Return value display      |
| Execution window          |
| Multi-Input window        |

+-------------------------------------------------------------+
You will first need to select the type of window you will create. You will then be asked to enter an 8-character name and an optional 40-character description. These are for your use only; they do not appear in the window during execution.

For a Vertical Menu, Horizontal Menu, Text Input, Text Display, File Names, Field Names, File Contents, Multi-Input, or Return Value Display window, you are prompted to supply a 60-character heading.

For a Text Input window, you are prompted to choose the format of the text entry field (alphanumeric, with all text translated to uppercase; alphanumeric, with no case translation; or numeric). Later, in the Window Design Screen, you can make the length of the text entry field shorter than the window’s header length by typing a single character in the window immediately following the last desired field position, or by typing characters continuously from the first field position to the last desired field position.

For a File Names, Field Names, or File Contents window, you are prompted to produce file-identification criteria that can consist of an amper variable, a complete file identification, or (for File Names windows) a file specification which includes an asterisk (for example, *MASTER).

The asterisk is used as a wildcard character: it indicates that any character or sequence of characters can occupy that position. In CMS, an asterisk used in file-identification criteria can be embedded (for example, *DEPT FOCEXEC); the asterisk can be used in the file name, the file type, and the file mode. In MVS, the asterisk can be used as the member name but not in the ddname.

If an amper variable is used, you can prompt for the file identification criteria at run time.

- File-identification criteria in CMS must specify the file name first, the file type second, and an optional file mode third. If the file mode is not specified, it defaults to an asterisk.

- File-identification criteria in MVS must specify the member name first and the ddname second.

If you are creating a Field Names window, your file-identification criterion is the name of a Master File.

In addition, you can create Execution windows containing FOCUS commands such as Dialogue Manager statements or TABLE requests. You will be prompted for the window name and heading. Once a window has been specified, you will see the Window Design screen.

For complete information about the types of windows you can create in Window Painter, see Types of Windows You Can Create on page 3-4.

The next screen displayed is the Window Design Screen, discussed in Window Design Screen on page 3-54. This screen enables you to enter information, and position and size your window.
Window Design Screen

In this screen you design the appearance and functionality of your windows. It appears during the window creation process, when you press Enter after typing the heading of your window.

The Window Design Screen consists of a blank screen, a cursor, and text asking you to move the cursor to the starting position for the window. This starting position becomes the upper left corner of the window. Use the cursor arrow keys to move the cursor to the place where you want the upper left corner of the window to be, and press Enter.

When you press Enter this time, the window appears, with its heading at the top. You can enlarge it, type text in it, and move it around the screen.

The Window Design Screen lets you use the keyboard to manipulate the window you are creating.
The following chart summarizes Window Design Screen key functions in all window types.

<table>
<thead>
<tr>
<th>PF Key</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF1</td>
<td>Displays a window of help information.</td>
</tr>
<tr>
<td>PF2</td>
<td>Displays the Window Options menu. This menu is discussed in <em>Window Options Menu</em> on page 3-56.</td>
</tr>
</tbody>
</table>
| PF3    | Displays the exit menu. You can select:  
Exiting from the Window Design Screen while saving your work.  
Quitting from the Screen without saving your work.  
Continuing your work. |
| PF4    | Resizes the window. First move the cursor to the desired position of the window’s lower right corner. When you press PF4, the window’s upper left corner remains in the same position; the window’s lower right corner moves to the current cursor position.  
If the window size is reduced, nothing in the window is deleted; all window contents beyond the window border can be seen by scrolling the window. |
| PF5    | Gets the GOTO value, the Return value and the FOCEXEC name for the active window. |
| PF6    | Sets the return value of the line which the cursor is on. |
| PF7    | Scrolls the window up if the window contents extend beyond the top border. |
| PF8    | Scrolls the window down if the window contents extend beyond the bottom border. |
| PF9    | Moves the window. First move the cursor to the desired position of the window’s upper left corner. When you press PF9, the window’s upper left corner (the + in the border) moves to the current cursor position. The rest of the window moves accordingly. |
| PF10   | Deletes the line of window contents identified by the current cursor position. If the window contents do not extend beyond the window borders, then the window itself will be reduced by one line. |
| PF11   | Adds one line of window contents beneath the line identified by the current cursor position. If the window contents do not extend beyond the window borders, then the window itself will increase by one line. |
PF Key | Function
--- | ---
PF12 | Provides the same function as the PF3 key.
PF13 - PF24 | These keys provide the same functions as the corresponding keys PF1 - PF12.

If a window’s contents extend beyond a top or bottom border, then the message (MORE) is displayed on that border. This reminds you that there are more lines of contents which are hidden beyond that border. You can view these lines by scrolling the window toward the border. When the window is used in an application, the user can also scroll the window to see all of the contents.

The display line at the bottom of the Window Design Screen shows instructions or information. When you first see the Window Design Screen, the display line tells you to move the cursor and press Enter. When you press Enter, the display line shows the name of the window file, and the name and type of window being created; it also tells which keys to press for the HELP function, the SIZE function, and the Window Options Menu.

**Window Options Menu**

When the Window Design Screen is displayed, pressing PF2 brings up the following Window Options Menu:

```
Would you like to:
Create a report?
Create a graph?
Exit?
```

Wind: MAIN  Typ: Menu (vert)  PF1=Help  2=Menu  4=Size  9=Move  10=Del  11=Add
The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goto value</td>
<td>Selecting this option lets you specify the next window in the path from this selection field or window. You will be asked to supply the name of the window. (It does not matter whether or not this window exists. You can create it later, just remember what you have called it.) In menu windows, goto values are assigned to each menu item. In other windows, there is a single goto value for the entire window. To assign a goto value, your cursor must be on the proper line when the Window Options Menu is brought up. Select Goto value from the Window Options Menu and you will be prompted to enter the name of the window that is the target of the goto. Type the name in the space provided and press Enter again. The goto value is assigned.</td>
</tr>
</tbody>
</table>

```
$EXETYPE $;
| Create a report? |
| Create a graph? |
| Exit? |
```

```
Wind: MAIN  Typ: Menu (vert)  F1=Help  2=Menu  4=Size  9=Move  10=Del  11=Add
```
### Menu Option | Description
---|---
**Return value** | The return value supplies a value for an amper variable. If the user selects this field during execution, the return value you have assigned is plugged into the amper variable in your FOCEXEC. Return values are assigned to each menu item in menu windows, and one per window for other window types. The only exceptions are the Multi-Input window, whose return value is the name of the input field occupied by the cursor when you pressed Enter or a PF key, and the Return Value Display window, which does not have a return value but instead displays other windows’ return values. The return value for a Multi-Select window is the number of selections.

To assign a return value, your cursor must be on the proper line when the Window Options Menu is brought up. Select Return value from the Window Options Menu and you will be prompted to enter a return value. Note that for File Names, Field Names, and File Contents windows, the value that you enter is the file-identification criterion for that window. Type the value in the space provided and press Enter again. The return value is assigned.
<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOCEXEC name</td>
<td>Attaches a FOCEXEC to each menu selection of the window. The FOCEXEC is executed when the menu item is selected.</td>
</tr>
<tr>
<td>Heading</td>
<td>Changes the heading of any window you are working on. You can also add or remove a heading.</td>
</tr>
<tr>
<td>Description</td>
<td>Changes the description of any window you are working on.</td>
</tr>
<tr>
<td>Show a window</td>
<td>Used only during window editing, brings another window onto the screen for reference. You cannot edit the second window.</td>
</tr>
<tr>
<td>Unshow a window</td>
<td>Removes the shown window from the display.</td>
</tr>
<tr>
<td>Display list</td>
<td>Enables you to specify a list of up to 16 windows that will be visible when this window is displayed during execution.</td>
</tr>
<tr>
<td></td>
<td>Note that if part of a window on the display list extends beyond the window border or does not fit on the screen, it cannot be scrolled.</td>
</tr>
<tr>
<td></td>
<td>As many as 16 windows can be displayed on the screen at one time. This applies to all windows on the screen (that is, a window displayed</td>
</tr>
<tr>
<td></td>
<td>during execution, windows displayed when executed previously and not hidden afterward, and windows displayed because specified on a</td>
</tr>
<tr>
<td></td>
<td>display list). The window facility interprets each window heading as a separate window: if all of the windows have headings, 16 of</td>
</tr>
<tr>
<td></td>
<td>them can be displayed on the screen at one time.</td>
</tr>
</tbody>
</table>
Designing Windows With Window Painter

Menu Option Description

Hide list

Allows you to specify windows that will not appear when this window is displayed during execution. You can specify up to 16 specific windows or all windows in the window file. If you select “All,” all the windows will be hidden except those in the display list. -- If you do not hide a window which was displayed, it will remain on the screen until another window which includes it on a hide list is displayed during execution.
## Window Painter Screens

- **Menu Option**: Popup (Off/On)
- **Description**: Makes the window disappear when the user presses Enter during execution. Defaults to OFF, which leaves the window on screen. Set Popup to OFF with text display windows as they do not work even if set to ON.
### Menu Option | Description
---|---
Help window | Lets you display information about a window or a menu item when a user presses PF1 (the Window facility HELP key) during execution. The information displayed is text within a specified Help window.

Note that if the PFKEY option is specified in the -WINDOW statement, you will have to explicitly set a PF key as the HELP key, as described in *Testing Function Key Values* on page 3-24.

When selecting the Help window option, you will be asked to supply the name of the Help window file which contains the Help window. Next, you will be asked to supply the name of the Help window itself. The Help window can be an existing window, or one that you will create.

If the Help window displays field names, it qualifies duplicates with the segment name.

You can use any window type for a Help window. A Text Display window is easiest, except when you want to supply different help information for each item in a Vertical Menu, Horizontal Menu (that is, item-specific help).

If you wish to assign item-specific help, use a File Contents window that displays a file containing text in the following format:

```plaintext
=>HELPFILE
=> menu item
this is the Help message you want the user to see.

where:
=>
    Is entered with an equal sign (=) and a greater-than sign (>).

HELPFILE
    Must be uppercase.

menu item
    Is the exact text of the menu item. Any blank spaces which precede this text in the menu must also precede this text here in the Help file. Note that at least one blank space always precedes the menu item text in a Vertical Menu, Horizontal Menu, or Multi-Input Window.
```
For example, if the first three lines of a Vertical Menu are

1. Generate a sales report
2. Generate a stock report

and there are three blank spaces between the left border of the window and the beginning of the text, then the file containing help text could look like this:

=>HELPFILE
=>  1. Generate a sales report
This option displays a list of existing sales report requests, and lets you select one of these requests to execute.
=>  2. Generate a stock report
This option displays a list of existing stock report requests, and lets you select one of these requests to execute.

The lines immediately following the menu item text are displayed when the user positions the cursor on the menu item and presses PF1.

In some cases you may wish to assign topic-specific help, but you may want the help text for some of the topics to be contained in a separate file. In these cases, on the line following the menu item text, replace the help message with the file identification of the file containing that menu item’s help message.

In CMS, use this file-identification format:

FILENAME= filename filetype [filemode]

In MVS, use this file-identification format:

FILENAME= membername ddname

To assign one set of instructions that can be used for multiple menu items, use the following syntax:

=>DEFAULT
This text appears when you have not written topic-specific help.

The DEFAULT text must be the last section in the Help file.

Lines beginning with an * are comment lines that are not displayed.
What follows is an example of a topic-specific Help file for the Main Menu used in the tutorial.

->HELPFILE
*Help file for tutorial/Main Menu
-> Create a report?
Choose this option if you wish to create a new report.
-> Create a graph?
Select this option if you wish to create pie charts, bar charts or other graphics.
-> Exit?
If you wish to leave the application, choose this option.

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line-break</td>
<td>Formats the contents of the Return Value Display window. This option is set when designing the windows from which you collect the return value(s) to be displayed.</td>
</tr>
<tr>
<td></td>
<td>When you select this option, you will see:</td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>New line before value</td>
</tr>
<tr>
<td></td>
<td>New line after value</td>
</tr>
<tr>
<td></td>
<td>Both</td>
</tr>
<tr>
<td></td>
<td>where:</td>
</tr>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Places return value directly after preceding value. If there is not enough room on this line, return value is placed on the next line.</td>
</tr>
<tr>
<td></td>
<td>New line before value</td>
</tr>
<tr>
<td></td>
<td>Places return value on the next line.</td>
</tr>
<tr>
<td></td>
<td>New line after value</td>
</tr>
<tr>
<td></td>
<td>Places return value on the same line as preceding value. Places next return value on next line.</td>
</tr>
<tr>
<td></td>
<td>Both</td>
</tr>
<tr>
<td></td>
<td>Places return value on a line by itself.</td>
</tr>
</tbody>
</table>
## Menu Option Description

<table>
<thead>
<tr>
<th><strong>Menu Option</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
</table>
| **Multi-Select** | Enables you to select multiple items from one window. The number of items you select is collected as the return value from that window; each selected item’s return value is stored in a temporary file in memory. You can later retrieve these stored values for use in a FOCEXC. Values for up to eight windows can be stored at one time.  

When you select this option, you will see:  

`Multi-Select (On )`  

During execution, the user selects individual values by pressing PF9. After all selections have been made, the user presses Enter.  

Note that when the `-WINDOW` command is issued with the PFKEY option, the PF9 key cannot be used to make selections unless a SET command is issued before the `-WINDOW` command. For example:  

`SET PF09=SELECT`  

You can also set a different PF key for selecting multiple items.  

A Multi-Select window can have no more than one goto value. Although in a Vertical Menu window you can assign a different goto value to each menu item, only the value assigned to the first item is effective.  

The return value collected for a window using the Multi-Select option is the number of values selected by the user.  

To retrieve the individual values, issue a special WINDOW call, as follows:  

`-WINDOW windowfile windowname GETHOLD`  

where:  

`windowfile`  

Is the name of the window file.  

`windowname`  

Is the name of the Multi-Select window.  

GETHOLD  

Is the special parameter that retrieves one value at a time from the temporary file.  

The value is assigned to the variable `&windowname`. |
Designing Windows With Window Painter

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu Option</td>
<td>The GETHOLD option requires at least two -WINDOW commands in your FOCEXEC. The first -WINDOW command (without the GETHOLD option) transfers control to the Window facility where a Multi-Select window is used. The second and subsequent -WINDOW commands use the GETHOLD option to retrieve the stored amper variables collected in a particular Multi-Select window. For each value to be retrieved, you will need a -WINDOW command with the GETHOLD option. Each value will be stored in &amp;windowname. If you wish to use this value, we recommend assigning it to another variable. For example, if the return value has the value 4, you would issue the special -WINDOW statement four times; each time you would collect the value from &amp;windowname. Alternatively, you could perform a loop. Note that -WINDOW with the GETHOLD option will not transfer control from the FOCEXEC to the Window facility.</td>
</tr>
<tr>
<td>Quit</td>
<td>Returns you to the Window Painter Entry Menu.</td>
</tr>
<tr>
<td>Input fields</td>
<td>Input fields pertain to Multi Input Windows. Selecting the field takes you to that field.</td>
</tr>
<tr>
<td>Menu text</td>
<td>Specifies a line of descriptive text, up to 60 characters long, for items on a Horizontal menu. Use the Text line option to position the text.</td>
</tr>
<tr>
<td>Text line (x+1)</td>
<td>On a Horizontal menu, positions descriptive text one or two lines above or below the menu. Valid values are x+1 or x+2 to place the text above the Horizontal menu, x-1 or x-2 to place the text below the Horizontal menu. Use the Menu text option to define the descriptive text.</td>
</tr>
<tr>
<td>Pulldown (off/on)</td>
<td>If the setting is ON, placing the cursor on an item in a Horizontal menu can display an associated Pulldown menu. The default setting is OFF. Turn the setting ON by positioning the cursor on this option and pressing Enter. — The Pulldown menu must be a Vertical menu and must be assigned as the goto value for the Horizontal menu item. Note that setting Pulldown ON automatically shuts off Menu Text.</td>
</tr>
<tr>
<td>Switch window</td>
<td>Enables you to work on and move between two windows. When you select this option, you can create a new window, or edit an existing window without returning to the Main Menu.</td>
</tr>
</tbody>
</table>
Utilities Menu

If you select the Utilities option from the Window Painter Main Menu, the Utilities Menu will be displayed:

<table>
<thead>
<tr>
<th>INSTRUCTIONS</th>
<th>Move cursor to selection and hit ENTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use PF3 or PF12 to undo a selection</td>
</tr>
<tr>
<td></td>
<td>Use PF1 for help</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Select one of the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document the file</td>
</tr>
<tr>
<td>Change the file description</td>
</tr>
<tr>
<td>Compress the file</td>
</tr>
<tr>
<td>Rename a window</td>
</tr>
<tr>
<td>Copy a window</td>
</tr>
<tr>
<td>Select the start window</td>
</tr>
<tr>
<td>Create a transfer file</td>
</tr>
<tr>
<td>Quit the Utilities Menu</td>
</tr>
</tbody>
</table>

File: SAMPLE  FOCUS WINDOW PAINTER
The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Document the file</strong></td>
<td>When you select this utility, Window Painter creates documentation of the window file. You can display the document on the screen using TED or another system editor, or send it to a printer or disk file.</td>
</tr>
<tr>
<td></td>
<td>In CMS, this option creates a file with file type TRF on your A disk.</td>
</tr>
<tr>
<td></td>
<td>In MVS, this option creates a member of the TRF PDS; that PDS must have already been allocated. However, creating a PDS is not necessary if you are only going to use the documentation file during the current FOCUS session: Window Painter will temporarily allocate the PDS.</td>
</tr>
<tr>
<td></td>
<td>This document contains detailed information about all the windows in the window file. It shows you the kinds of windows, their structure and format, and any options you have assigned from the Window Options Menu, including return and goto values. The text you enter when prompted for a window file description or individual window description is part of this document.</td>
</tr>
<tr>
<td></td>
<td>The document is especially useful when creating a FOCEXEC, since it provides return and goto values in addition to other information.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you create another file with the same name, the file is not overwritten. It is appended.</td>
</tr>
</tbody>
</table>

```
-# WINDOW FILE NAME=SAMPLE
-# DESCRIPTION='Sample file for windows tutorial'
-# WINDOW NAME=MAIN, TYPE=Menu (vertical)
-# DESCRIPTION='User can report, graph, or exit'
-# ROW= 6,COLUMN=23,HEIGHT= 7,WID=38,WINDOW= 7,POPUP= 0,BORDER= 2,HEADLEM=28,
-# RETURN=None
-# MULTI=Off
-# HEADING:
-# Would you like to:
-# WINDOW DATA: GOTOS: VALUES: 
-# 1. Exit? 'EXIT ', '' , 'EXIT '
-# 6. Exit? 'EXIT ', '' , 'EXIT '
-# DISPLAY LIST:
-# BORDER
```
<table>
<thead>
<tr>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change the file description</td>
<td>Changes the description of the current window.</td>
</tr>
<tr>
<td>Compress the file</td>
<td>This utility is provided to help you save space in memory. It allows space made available by deleted or edited windows to be reused.</td>
</tr>
<tr>
<td>Rename a window</td>
<td>When you select this utility, you see a list of the windows in the current window file. You can change the name of any of these windows.</td>
</tr>
</tbody>
</table>
| Copy a window               | This function copies a window from one window file to another, or duplicates it within the same file.  
                                  The copy function is useful when you create a new application, or need to add windows to an existing application, and want the windows to look like those you have already created. You can copy any window and edit it to conform to the new application. |
| Select the start window     | Enables you to choose a default start window. This window is the first to be entered if a specific window is not selected upon startup. If a default start window is not explicitly chosen, FOCUS will select the first window created to be the start window. |
| Create a transfer file      | Creates a file to be transferred for use with the Window facility in PC/FOCUS, TSO or another FOCUS environment.  
                                  In CMS, this option creates a file with file type TRF on your A disk.  
                                  In MVS, this option creates a member of the TRF PDS; that PDS must have already been allocated. |
| Quit the utilities menu     | Returns you to the Main Menu.                                               |
Transferring Window Files

If you use FOCUS in more than one operating environment, you can transfer an existing window file from one environment to be used in another environment. For example, if you have a fully-developed window application in PC/FOCUS, and you want to develop a similar application in mainframe FOCUS, you can transfer the PC/FOCUS window file to mainframe FOCUS; this saves you the trouble of recreating the window file from scratch in mainframe FOCUS.

You can transfer a window file to a new environment in four simple steps:
1. Create a transfer file from the original window file using Window Painter.
2. Transfer the new file to the new environment using the XFER command.
3. Edit the transferred file in TED, if necessary.
4. Compile the transferred file using the WINDOW COMPILE command.

These steps are described in the following sections.

Creating a Transfer File

The window files which you design in Window Painter are compiled files; before a window file can be transferred to another environment, a user-readable source code version must be created. This user-readable file is called a transfer file, and is created using the transfer file option of Window Painter.

- In CMS, this Window Painter option automatically creates a transfer file with a file type of TRF on your A disk.

- In MVS, this Window Painter option automatically creates a new member of the PDS allocated to ddname TRF; the PDS must already have been allocated (with LRECL between 80 and 132 and RECFM FB). However, it is not necessary to create the PDS if you are only going to use the transfer file during the current FOCUS session: Window Painter will temporarily allocate the PDS.

- For information about the transfer files created by FOCUS Window Painter in other operating environments, see the appropriate FOCUS Users Manual for those environments.

To convert a window file to a transfer file, go to the Window Painter Utilities Menu and select:

Create a transfer file

You will then be prompted for the name of the new transfer file. Enter any name that you wish; it can have the same name as the window file, or an entirely new name. In CMS the name that you enter is the file name; in MVS it is the member name.
Transferring Window Files

Note that you should not give the transfer file a name already assigned to a window documentation file. Also, you should not give the transfer file a name already assigned to an existing transfer file unless you want to merge the two files, as described below. See the appropriate operating environment chapter in the *Overview and Operating Environments* manual for more information about duplicate window transfer and window documentation file names.

You will be asked to select which window(s) you want to transfer. You can select All to transfer all of the windows in the current window file, or you can select any single window in the file. This is the last step in creating a transfer file.

Note that you can merge transfer files: if a transfer file already exists for your window file, and you only need to add a new window to it, you can give the new transfer file the same name as the old one, and then select the new window. Window Painter will merge the source code for the new window into the existing file, so that you have a single complete transfer file.

**Transferring the File to the New Environment**

Once the transfer file exists, it can be transferred to the new environment using the XFER command. The XFER command is described in Chapter 7, *Enhancing Application Performance*.

**Editing the Transfer File**

Window facility features introduced in one FOCUS release may not be fully supported in earlier releases. Because different operating environments may be running different releases of FOCUS, the transfer file created by the FOCUS Window facility in one environment may contain features not fully supported by the Window facility in another environment.

If your transfer file contains Window facility features not fully supported in the new environment, you may need to remove or fine-tune those features. If, on the other hand, the new environment supports features not supported in the original environment, you can add those features to the transfer file. Adding, removing, and fine-tuning features can be done by simply editing the transfer file.
The Format of the Transfer File

The transfer file is a user-readable source code listing of all of the windows, and their features, that were included from the original window file. You can remove or fine tune an unsupported feature by simply editing or deleting the appropriate line in the transfer file; this can be done by using TED or any other editor.

Each transfer file contains:

- One set of window file attributes describing the file.
- For each window defined in the file, one set of window attributes describing that window.
- For each line in each window, one set of attributes describing that line.

If any attribute is not specified in the transfer file, it defaults to a value of zero or blank (depending on whether the value is normally numeric or alphanumric).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILENAME</td>
<td>The name of the original window file.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>A comment field describing the file.</td>
</tr>
<tr>
<td>WINOWNAME</td>
<td>The name of the window.</td>
</tr>
<tr>
<td>TYPE</td>
<td>The type of window:</td>
</tr>
<tr>
<td></td>
<td>1. Vertical Menu</td>
</tr>
<tr>
<td></td>
<td>2. Text Input Window</td>
</tr>
<tr>
<td></td>
<td>3. Text Display Window</td>
</tr>
<tr>
<td></td>
<td>4. Horizontal Menu</td>
</tr>
<tr>
<td></td>
<td>5. File Names Window</td>
</tr>
<tr>
<td></td>
<td>6. Field Names Window</td>
</tr>
<tr>
<td></td>
<td>7. File Contents Window</td>
</tr>
<tr>
<td></td>
<td>8. Return Value Display Window</td>
</tr>
<tr>
<td></td>
<td>9. Execution Window</td>
</tr>
<tr>
<td></td>
<td>10. Multi-Input Window</td>
</tr>
<tr>
<td>COMMENT</td>
<td>A comment field describing the window.</td>
</tr>
</tbody>
</table>
## Attribute Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRANSLATE</strong></td>
<td>Type of input for Text Input windows (Type 2).</td>
</tr>
<tr>
<td>0</td>
<td>Allow mixed case input.</td>
</tr>
<tr>
<td>1</td>
<td>Allow numeric input only.</td>
</tr>
<tr>
<td>2</td>
<td>Translate input to uppercase.</td>
</tr>
<tr>
<td><strong>ROW</strong></td>
<td>The row number of the upper left corner of the window.</td>
</tr>
<tr>
<td><strong>COLUMN</strong></td>
<td>The column number of the upper left corner of the window.</td>
</tr>
<tr>
<td><strong>HEIGHT</strong></td>
<td>The height of the window data (the number of lines of window data, not the height of the actual window frame). If there are more data lines than will fit in the window frame, the PF7 and PF8 keys can scroll the window.</td>
</tr>
<tr>
<td><strong>TEXT LINE</strong></td>
<td>Position of menu text. Values are: +1, +2, -1, -2.</td>
</tr>
<tr>
<td><strong>WIDTH</strong></td>
<td>The width of the window frame, not including the border.</td>
</tr>
<tr>
<td><strong>INPUT FIELDS</strong></td>
<td>Fields for Multi-Input Windows.</td>
</tr>
<tr>
<td><strong>WINDOW</strong></td>
<td>The number of lines in the actual window frame (not the number of lines of window data). This does not include borders.</td>
</tr>
<tr>
<td><strong>POPUP</strong></td>
<td>Sets the pop-up feature.</td>
</tr>
<tr>
<td>0</td>
<td>This will not be a pop-up window.</td>
</tr>
<tr>
<td>1</td>
<td>This will be a pop-up window.</td>
</tr>
</tbody>
</table>

*Figure 3-1. Transfer File Syntax: Window File Attributes*
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORDER</td>
<td>Sets the window border.</td>
</tr>
<tr>
<td>0</td>
<td>There will be no window border.</td>
</tr>
<tr>
<td>1</td>
<td>There will be a window border.</td>
</tr>
<tr>
<td>2</td>
<td>There will be a window border.</td>
</tr>
<tr>
<td>Options 1 and 2 both result in a basic window border.</td>
<td></td>
</tr>
<tr>
<td>HEADLEN</td>
<td>Length of the window heading. If this value is 0, there will be no heading.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Sets the line break feature for use with Return Value Display windows.</td>
</tr>
<tr>
<td>0</td>
<td>Line break will not be used.</td>
</tr>
<tr>
<td>1</td>
<td>New line before this return value.</td>
</tr>
<tr>
<td>2</td>
<td>New line after this return value.</td>
</tr>
<tr>
<td>3</td>
<td>New line before and after this value.</td>
</tr>
<tr>
<td>MULTI</td>
<td>Sets the multi-select feature.</td>
</tr>
<tr>
<td>0</td>
<td>This will not be a multi-select window.</td>
</tr>
<tr>
<td>1</td>
<td>This will be a multi-select window.</td>
</tr>
<tr>
<td>HEADING</td>
<td>The text of the window heading.</td>
</tr>
<tr>
<td>HELP</td>
<td>The name of the help window for this window.</td>
</tr>
<tr>
<td>HELPFILE</td>
<td>The name of the window file which contains the help window.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>The name of a window to be displayed at the same time this one is displayed. There can be up to 16 DISPLAY values for each window. This attribute is optional.</td>
</tr>
<tr>
<td>HIDE</td>
<td>The name of a window to be hidden when this one is displayed. There can be up to 16 HIDE values for each window. This attribute is optional.</td>
</tr>
</tbody>
</table>

*Figure 3-2. Transfer File Syntax: Window Attributes*
Table of Contents

Transferring Window Files

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>A line to be displayed in the window (for example, a menu choice in a Vertical Menu Window, or a line of text in a Text Display Window). The data can include amper variables (including &amp;windowname).</td>
</tr>
<tr>
<td>GOTO</td>
<td>The name of the window to go to if this line is selected by the user. The value can be an amper variable (including &amp;windowname). If the value is blank, and this line is selected, Windows will return to Dialogue Manager.</td>
</tr>
<tr>
<td>VALUE</td>
<td>The return value supplied if this line is selected by the user. This value will be placed in the amper variable &amp;windowname, where windowname is the name of the window. For File Names Windows (TYPE = 5), this is the file selection criteria (including asterisks) of the file names to be displayed. For Field Names Windows (TYPE = 6), this is the name of the Master File whose fields will be displayed. For File Contents Windows (TYPE = 7), this is the name of the file whose contents are to be displayed.</td>
</tr>
</tbody>
</table>

Figure 3-3. Transfer File Syntax: Window Line Attributes

Operating Environment Considerations

When you transfer a window file to a mainframe operating environment from a different environment, differences in hardware and operating software may require that you make changes to the file. These changes are discussed below.

- **Screen position.** Windows should not begin in row 1 or in column 1. If you transfer a window with these row or column positions, truncation will occur. Adjust the ROW and COLUMN attributes if necessary.
- **Screen size.** Windows should not have more than 22 rows or 77 columns. Windows which extend beyond the end of the terminal screen will automatically be truncated without any warning message.

This is important to note if you are transferring a window file from an environment where the screen size differs from that in the mainframe environment. Adjust the ROW and COLUMN attributes if necessary.

- **Window Position.** Column 1 of Vertical Menu, Horizontal Menu, Multi-Input and Text Display windows cannot be used. Window text must begin to the right of column 1.
- **Function keys.** Windows transferred from other environments may refer to function keys not present in the mainframe environment. Change function key references if necessary.
Designing Windows With Window Painter

- **Blank lines.** Are acknowledged by Window Painter.
- **Colors and Border Types.** The use of colored windows and background and multiple border types is not supported.
- **File Naming Conventions.** File naming conventions differ in different operating environments. When transferring a file from some environments, the Window facility will automatically translate references to FOCEXECs, Master Files, and error files, as shown below. You must change other file references yourself when you edit the transfer file.

<table>
<thead>
<tr>
<th>PC or UNIX Extension</th>
<th>Mainframe File Type or ddname</th>
</tr>
</thead>
<tbody>
<tr>
<td>.FEX</td>
<td>FOCEXEC</td>
</tr>
<tr>
<td>.MAS</td>
<td>MASTER</td>
</tr>
<tr>
<td>.ERR</td>
<td>ERRORS</td>
</tr>
</tbody>
</table>

**Example**  

**Sample Transfer File**

To illustrate the transfer file format, part of the transfer file for the SAMPLE window file is shown below (SAMPLE is described in the tutorial). The MAIN and EXECNAME windows from the file are included in the example.

```plaintext
FILENAME=SAMPLE
DESCRIPTION='Sample file for windows tutorial'
WINDOWNAME=MAIN,TYPE=1
COMMENT='User can report, graph, or exit.'
ROW=6,COLUMN=23,HEIGHT=7,WIDTH=38,WINDOW=7,POPUP=0,BORDER=2,HEADLEN=28
RETURN=0
MULTI=0
HEADING='Would you like to:'
DATA=''
$ DATA='Create a report?'
GOTO='EXECTYPE',VALUE='RPT'
$ DATA=''
$ DATA='Create a graph?'
GOTO='EXECTYPE',VALUE='GRPH'
$ DATA=''
$ DATA='Exit?'
GOTO=' ',VALUE='XXIT'
$ DATA=''
$ DISPLAY=BORDER,$
DISPLAY=BANNER,$
WINDOWNAME=EXECNAME,TYPE=5
```
Transferring Window Files

Compiling the Transfer File

The transfer file can be executed in its current format, but it may execute slowly, and it will use a large amount of memory. You can make your window application more efficient, requiring less time and memory for execution, by compiling it.

You can compile a transfer file using the WINDOW COMPIL command. This produces a new compiled window file, in the same format as the window files produced by Window Painter.

Note that before you can issue this command in MVS, a PDS with LRECL 4096 and RECFM F must have already been allocated to ddname FMU. However, you do not need to create this PDS if you are only going to use the transfer file during the current FOCUS session: Window Painter will temporarily allocate the PDS.
Syntax How to Compile a Transfer File

The command syntax is

WINDOW COMPILE windowfile

where:

windowfile

Is the name of the transfer file.

In CMS, this must be the file name of a file with file type TRF.

The command will create a new file with the file name specified in the command, and a file type of FMU, on the A disk. Once it has been created, you can move the file to any disk you wish.

In MVS, this must be a member name of a member of a PDS allocated to ddname TRF. The command will create a new member of the PDS allocated to ddname FMU, with the same member name specified in the command.

When a Dialogue Manager -WINDOW statement is encountered in a FOCEXEC, FOCUS will search for a compiled window file (an FMU file) with the specified file name. If the compiled file is not found, the transfer file (TRF file) with the same file name will be used.

Note that if you compile a transfer file and later make changes to it, you will need to recompile the updated transfer file; otherwise, FOCUS will continue to use the older, unchanged compiled file.
4 Setting Parameters: SET

The SET command enables you to change parameters that govern your FOCUS environment. These parameters control output, work areas, the Hot Screen facility and other FOCUS features.

Syntax How to Set Parameters

The syntax is

SET parameter = option[, parameter = option,...]

where:

parameter
  Is the FOCUS setting you wish to change.

option
  Is one of a number of options available for each parameter.

You can set several parameters in one command by separating each with a comma. You may include as many parameters as you can fit on one line. Repeat the SET keyword for each new line.

Syntax How to Set Parameters in a Request

Many SET commands that change system defaults can be issued from within TABLE and GRAPH requests. SET used in this manner is temporary, affecting only the current request. The syntax is

ON {TABLE|GRAPH} SET parameter value [AND parameter value ...]

where:

parameter
  Is the system default you wish to change.

value
  Is an acceptable value that will replace the default value.
Setting Parameters: SET

**Example** Setting Parameters in a Request

For example,

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID
ACROSS DEPARTMENT
ON TABLE SET NODATA NONE
END
```

changes the default NODATA character for missing data from a period to the word NONE.

SET commands that cannot be issued from within TABLE include ASNAMES, BINS, and HOLDATTR.

**Reference** Parameters Controlled by the SET Command

This section lists, alphabetically, general parameters controlled by the SET command.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCBLN</td>
<td>ON, OFF</td>
<td>Accepts blank or zero values for fields with ACCEPT statements in the Master File (see the Describing Data manual). If ACCBLN is set to OFF, blank and zero values are not accepted for fields with ACCEPT statements unless blank or zero values are explicitly coded in the list of acceptable values. ON is the default.</td>
</tr>
<tr>
<td>AGGRRAIOT</td>
<td>n</td>
<td>Ratio of aggregation based on retrieved records and final size of answer set.</td>
</tr>
<tr>
<td>AGGR</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
| ALL       | ON, OFF, PASS | Handles missing segment instances in reports. The settings are:

  - **ON**: Missing segment instances appear in reports when fields in the segment are not screened by WHERE or IF statements in the requests. The missing field values are denoted by the NODATA character you set with the NODATA parameter (see NODATA).
  - **OFF**: Missing segment instances do not appear in reports. OFF is the default.
  - **PASS**: Missing segment instances appear in reports regardless of IF statements in the requests. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOWCVTERR</td>
<td>ON</td>
<td>Allows the display of a record containing an incorrect date. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ON</strong></td>
<td>Allows the display of a field containing an incorrect date.</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td>Generates an error if bad data is encountered and does not display the record containing the bad data.</td>
</tr>
<tr>
<td>ASNAMES</td>
<td>ON</td>
<td>Controls the FIELDNAME attribute in a HOLD Master File. When an AS phrase is used in a TABLE request, the literal specified can be used as the field name in a HOLD file. Also controls how field names are specified for the values of an ACROSS field when a HOLD file is created. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>ON</strong></td>
<td>Uses the AS phrase for the field name and controls the way ACROSS fields are named in HOLD files in any format.</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td>Does not use the AS phrase for a field name in HOLD files or affect the way ACROSS fields are named.</td>
</tr>
<tr>
<td></td>
<td><strong>FOCUS</strong></td>
<td>Uses the AS phrase for the field name and controls the way ACROSS fields are named in HOLD files only in FOCUS format. FOCUS is the default.</td>
</tr>
</tbody>
</table>
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOINDEX</td>
<td>ON&lt;br&gt;OFF</td>
<td>Retrieves data faster by automatically taking advantage of indexed fields in most cases where TABLE requests contain equality or range tests on those fields in FOCUS files. The settings are:&lt;br&gt;&lt;br&gt;<strong>ON</strong>&lt;br&gt;Uses indexed retrieval when possible.&lt;br&gt;&lt;br&gt;<strong>OFF</strong>&lt;br&gt;Uses indexed retrieval only when explicitly specified via an indexed view. OFF is the default.&lt;br&gt;&lt;br&gt;Note:&lt;br&gt;- AUTOINDEX is never performed when the TABLE request contains an alternate file view. For example, TABLE FILE filename.fieldname.&lt;br&gt;- Indexed retrieval is not performed when the TABLE request contains BY HIGHEST or BY LOWEST phrases and AUTOINDEX is ON. Indexed retrieval is only performed when the request uses an indexed view.</td>
</tr>
<tr>
<td>AUTOPATH</td>
<td>ON&lt;br&gt;OFF</td>
<td>Dynamically selects an optimal retrieval path for accessing a FOCUS file by analyzing the fields referenced and the file structure, and choosing the lowest possible segment as its entry point. Will use AUTOPATH only if your selection field is not indexed. The settings are:&lt;br&gt;&lt;br&gt;<strong>ON</strong>&lt;br&gt;Turns AUTOPATH on. ON is the default.&lt;br&gt;&lt;br&gt;<strong>OFF</strong>&lt;br&gt;Turns AUTOPATH OFF.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Options</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AUTOSTRATEGY</td>
<td>ON</td>
<td>When searching a FOCUS file for the key field in the segment specified in a WHERE or IF test, this parameter tells FOCUS when to stop traversal. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ON</td>
<td>On: Stops traversal when past the value for the key field. ON is the default.</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>Off: Continues traversal when past the value for the key field.</td>
</tr>
<tr>
<td>AUTOTABLEF</td>
<td>ON</td>
<td>Avoids creating the internal matrix based on the features used in the query. Avoiding internal matrix creation reduces internal overhead costs and yields better performance.</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>BINS</td>
<td>$n$</td>
<td>Number of 4K pages in memory used for database buffers. You can vary BINS from 13 to 64 pages depending on the size of memory. The default is roughly two-thirds of the core remaining after you start FOCUS.</td>
</tr>
<tr>
<td>BLKCALC</td>
<td>NEW</td>
<td>For MVS only, enables system-determined blocking for HOLD files written to DASD; files written to tape have BLKSIZE 32760, the operating-system maximum. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NEW</td>
<td>New: Calculates optimal blocking factors for both 3380 and 3390 device types. NEW is the default.</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td>Old: Uses the method of calculating BLKSIZE that was used prior to FOCUS Release 6.8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The SET BLKCALC command must be issued before the TABLE request and cannot be set within a request.</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| BUSDAYS       | _MTWTF_       | Allows you to specify which days are considered business days and which days are not if your business does not follow the traditional Monday through Friday week.  

_MTWTF_ represents the traditional work week, with underscores replacing Saturday and Sunday. Any day that you do not wish to designate as a business day must be replaced with an underscore in its designated space. |
| BYPANEL       | ON, OFF, 0, n | Controls the display of BY fields on the panels of a report. You can also specify the parameter as BYPANELING. The settings are:  

**ON**  
All BY fields specified in the report request display on each panel.  

**OFF**  
BY fields are displayed on the first panel but not for subsequent panels and columns may split between panels. OFF is the default.  

**0**  
BY fields are displayed on the first panel but not for subsequent panels; prevents column splitting.  

**n**  
The number of BY fields to be displayed; \( n \) is less than or equal to the total number of BY fields specified in the request, from the major sort (first BY field) down.  

**Note:** When BYPANEL is specified, the maximum number of panels is 99. When BYPANEL is OFF, the maximum number of panels is four. |
| BYSCROLL      | ON, OFF       | Allows you to make report headings and footers scroll along with the report contents in your Hot Screen report. |
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOTTOMMARGIN</td>
<td>value</td>
<td>Sets the bottom boundary of report contents on a page. The default value is .25 inches. <strong>Note:</strong> This command applies only to PostScript and PDF report formats.</td>
</tr>
<tr>
<td>CACHE</td>
<td>0</td>
<td>Stores 4K FOCUS database pages in memory and buffers them between the database and BINS. <strong>Note:</strong> This command applies only to PostScript and PDF report formats.</td>
</tr>
<tr>
<td></td>
<td>n</td>
<td>Allocates no space to cache; cache is inactive. The default is 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is the number of 4K pages of contiguous storage allocated to cache memory. The minimum is two pages; the maximum is determined by the amount of memory available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To clear the CACHE setting issue a SET CACHE=n command. This causes FOCUS to flush the buffer, that is, everything in cache memory is lost.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When a procedure calls for a read of a database page, FOCUS first searches BINS, then cache memory, and finally, the database on disk. If the page is found in cache, FOCUS does not have to perform an I/O to disk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When a procedure calls for a write of a database page, the page is written from BINS to disk. The updated page is also copied into cache memory so that the cache and disk versions remain the same. Unlike reads, cache memory does not save disk I/Os for write procedures.</td>
</tr>
</tbody>
</table>
Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACHE</td>
<td>𝑛</td>
<td>FOCSORT pages are also written to cache; when the cache becomes full, they are written to disk. For optimal results, set cache to hold the entire database plus the size of FOCSORT for the request. To estimate the size of FOCSORT for a given request, issue the ? STAT command (discussed in Chapter 6, FOCUS Query Commands); then add the number of SORTPAGES listed to the number of database pages in memory and issue a SET CACHE command for that amount. If cache is set to 50, 50 4K pages of contiguous storage are allocated to cache. The maximum number of cache pages can be set at installation. For more information, see Technical Memo 7838.1, Setting the Maximum Number of Cache Pages.</td>
</tr>
<tr>
<td>CARTESIAN</td>
<td>ON, OFF</td>
<td>Generates a report containing all combinations of non-related data instances in the case of a multi-path request. The settings are: ON Generates a report with non-related records. OFF Disables the Cartesian product. OFF is the default.</td>
</tr>
</tbody>
</table>
Continental Decimal Notation. This numerical notation uses a comma to mark the decimal position in a number, and periods (.) to mark off significant digits into groups of three.

**ON**
Indicates use of Continental Decimal Notation. For example, the number 3,045,000.76 is represented as 3.045.000,76.

**OFF**
Turns CDN off. For example, the number above is represented as 3,045,000.76. OFF is the default.

CDN is supported for data entry in MODIFY via FIDEL and in FSCAN and for output in TABLE requests. It is not supported for DEFINE or COMPUTE expressions or in MODIFY validation statements.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDN EROPE</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continental Decimal Notation. This numerical notation uses a comma to mark the decimal position in a number, and periods (.) to mark off significant digits into groups of three.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indicates use of Continental Decimal Notation. For example, the number 3,045,000.76 is represented as 3.045.000,76.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Turns CDN off. For example, the number above is represented as 3,045,000.76. OFF is the default.</td>
</tr>
<tr>
<td>COLUMNSCROLL</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enables you to scroll by column within the panels of a report provided that the report is wider than the screen width.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enables column scrolling to the right and left by pressing the PF10 key and the PF11 key, respectively. To scroll up and down within the same column, use the PF7 key and the PF8 keys.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disables column scrolling. OFF is the default.</td>
</tr>
<tr>
<td>COUNTWIDTH</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>LISTWIDTH</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expands the default format of COUNT fields from a five-byte integer to a nine-byte integer.</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>COMPUTE</strong></td>
<td>NEW</td>
<td>Compiles all COMPUTE calculations in DEFINE statements and MODIFY requests into machine code at request time; uses this code to perform calculations at run time.</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NEW</strong></td>
<td>Specifies the new, compiled logic. NEW is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>OLD</strong></td>
<td>Forces all calculations into the old logic until the FOCUS session is over, or the SET COMPUTE command is reset.</td>
</tr>
<tr>
<td><strong>DATEDISPLAY</strong></td>
<td>ON</td>
<td>Displays base dates in a FOCUS report. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>Displays blanks if the data is the base date value. OFF is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>ON</strong></td>
<td>Displays the base date if the data is the base date value.</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DATEDISPLAY</strong></td>
<td><strong>ON</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td></td>
</tr>
</tbody>
</table>

Previously, TABLE always displayed a blank when a date read from a file matched the base date or a field with a smart date format had the value 0. The following shows the base date for each supported date format:

<table>
<thead>
<tr>
<th>Format</th>
<th>Base Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>YMD and YYMD</td>
<td>1900/12/31</td>
</tr>
<tr>
<td>YM and YYM</td>
<td>1901/01</td>
</tr>
<tr>
<td>YQ and YYQ</td>
<td>1901/Q1</td>
</tr>
<tr>
<td>JUL and YYJUL</td>
<td>00/365 and 1900/365</td>
</tr>
</tbody>
</table>

**Note:** You cannot set DATEDISPLAY with the ON TABLE command.

**DATEFNS** | **ON**  | Loads the year 2000-compliant versions of the FUSELIB subroutines. Only set DATEFNS to OFF if you want to use the non-year 2000-compliant subroutines. |
|-----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------|
Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME</td>
<td>STARTUP, CURRENT</td>
<td>Sets time and date in reports. This command is useful for determining (statically or dynamically) exactly when your report was run. You can display the DATETIME value using any FOCUS date variable, for example, YMD, MDY, TOD, etc. If DATETIME is not set, the behavior of the FOCUS date variables remain the same.</td>
</tr>
<tr>
<td></td>
<td>NOW, RESET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The settings for DATE are:</td>
</tr>
<tr>
<td></td>
<td>STARTUP</td>
<td>Is the time and date when you began your FOCUS session. This setting is the default.</td>
</tr>
<tr>
<td></td>
<td>CURRENT</td>
<td>NOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CURRENT and NOW are interchangeable; you may use either term.</td>
</tr>
<tr>
<td></td>
<td>RESET</td>
<td>Freezes the date and time of the current run for the rest of the session or until another SET DATETIME command is issued.</td>
</tr>
<tr>
<td>DEFCENT</td>
<td>n, 19</td>
<td>Used to establish century digits in conjunction with YRTHRESH to define a 100-year window for the sliding window technique.</td>
</tr>
<tr>
<td>DFC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>OFF, PCCOLOR, PCMONO</td>
<td>Is the PC display mode selection. The settings are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCCOLOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PCMONO</td>
</tr>
</tbody>
</table>
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| ESTRECORDS   | $n$     | Used to pass the estimated number of records to be sorted in the request. FOCUS queries using external sorts and including the parameter ‘FILSZ=En’ can diminish FOC909 errors. This parameter enables the sorting algorithms to estimate SORTWORK space requirements for each sort parameter request. The setting is:  

\[
\text{ESTRECORDS} = n
\]

- $n$ is the estimated number of records to be sorted.

In order to make an accurate estimate for your ESTRECORDS setting, it is suggested that you run the report without an external sort in order to get a record count. If an attempt is made to SET ESTRECORDS from the FOCUS prompt, FOCPARM, or PROFILE FOCEXEC the following error is generated:

```
SET ESTRECORDS = n
(FOC36210) THE SPECIFIED PARAMETER CAN ONLY BE SET ON TABLE: ESTRECORDS
```

**Note:**

- ESTRECORDS can only be set with the ON TABLE SET command within the TABLE, MATCH, or GRAPH request.
- For CMS/SyncSort the ‘FILSZ=En’ parameter is ignored. Therefore, SET ESTRECORDS $n$ has no effect.

| EMPTYREPORT  | ON | OFF | Controls the output generated when a TABLE request retrieves zero records. The settings are:  
|--------------|----|-----|---------------------------------------------------------------------------------------------------------------------------------------------|
|              |    |     | ON  

- Generates an empty report when zero records are found.

OFF  

- Does not generate an empty report when zero records are found. OFF is the default.

EMPTYREPORT is not supported with TABLEF. When a TABLEF request retrieves zero records, an empty report is always generated.
### Parameter Options Explanation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTAGGR</td>
<td>ON, OFF, NOFLOAT</td>
<td>Enables you to use external sorts to perform aggregation. The settings are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong> Allows aggregation by an external sort.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Disallows aggregation by an external sort.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>NOFLOAT</strong> Allows aggregation if there are no floating data fields present.</td>
</tr>
<tr>
<td>EUROFILE</td>
<td>ddname, OFF</td>
<td>Brings the relevant currency database into memory. This setting is by default not set at all. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ddname</strong> Is the name of the Master File for the currency database. There is no default value for EUROFILE. The ddname must refer to a data source known to FOCUS and accessible by FOCUS in read-only mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Deactivates the currency database and removes it from memory.</td>
</tr>
<tr>
<td>EXTHOLD</td>
<td>ON, OFF</td>
<td>Enables you to create HOLD files using an external sort. The settings are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong> Enables HOLD files by an external sort.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Disables HOLD files by an external sort.</td>
</tr>
</tbody>
</table>

**Note:**
- You cannot append any additional SET parameters to the SET EUROFILE command line. For example, the PAUSE setting would be lost if you issued the following command:
  ```
  SET EUROFILE=filename, PAUSE=OFF
  ```
- You cannot issue the SET EUROFILE command within a TABLE request.
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTSORT</td>
<td><strong>ON</strong></td>
<td>Calls an external sort for use with the TABLE, MATCH, and GRAPH commands. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td><strong>ON</strong> Enables FOCUS to pass records that are retrieved to an external sort. ON is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Uses the FOCUS internal sorting procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the report can be processed entirely in memory, external sorting does not occur. In order to determine if the report can be processed in memory, issue the ? STAT query after the TABLE, MATCH, or GRAPH command, and check the value of the SORT USED parameter. For additional information, see the Creating Reports manual.</td>
</tr>
<tr>
<td>EXTTERM</td>
<td><strong>ON</strong></td>
<td>Enables use of extended terminal attributes. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td><strong>ON</strong> Enables use of attributes. ON is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Disables use of attributes.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Options</td>
<td>Explanation</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FIELDNAME</td>
<td>NEW</td>
<td>Increases the maximum length of field names and aliases to 66 characters. Field names and aliases may be qualified. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTRUNC</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NEW</strong></td>
<td>Lets you use field names of up to 66 characters; qualified field names are supported. NEW is the default.</td>
</tr>
<tr>
<td></td>
<td><strong>OLD</strong></td>
<td>Lets you use field names of up to 12 characters; qualified field names are not supported. The old maximum length of field names and aliases used with interface products may not be 12 characters. Refer to the appropriate Interface manual for further information.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTRUNC</strong></td>
<td>Does not allow unique truncations. Supports 66-character and qualified field names.</td>
</tr>
<tr>
<td></td>
<td><strong>NEW</strong></td>
<td>The maximum of 66 characters includes the name of the field or alias plus eight-character maximums for field qualifiers (Master File name and segment name) and delimiting characters (periods).</td>
</tr>
<tr>
<td></td>
<td><strong>OLD</strong></td>
<td>If you change the value of FIELDNAME within a FOCUS session from OLD to NEW or NOTRUNC, or from NEW or NOTRUNC to OLD, then existing JOINs, DEFINEs, and COMBINEs are cleared.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FILE[NAME]</th>
<th>filename</th>
<th>Default file name for FOCUS commands. When you set a default file name, you can use that file without specifying its name. For example:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>SET FILE= CAR</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>TABLE</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>PRINT CAR BY COUNTRY</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>END</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILTER</td>
<td>`*</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* Denotes all declared filters. This setting is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON Activates the filter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF Deactivates the filter. OFF is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xx Is the name of a filter as declared in the NAME = syntax of the FILTER FILE block.</td>
</tr>
<tr>
<td></td>
<td>Note:</td>
<td>• The maximum number of filters set ‘ON’ for a file is limited by the number of IF/WHERE statements in these filters and should not exceed the standard FOCUS limit of IF/WHERE statements in any single TABLE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The SET FILTER command is limited to one line. To activate more filters to fit on one line repeat the SET FILTER command.</td>
</tr>
<tr>
<td>FIXRETRIEVE</td>
<td>ON</td>
<td>FOCUS HOLD files support keyed retrieval, which can greatly reduce the IOs incurred in reading extract files. The performance gains are accomplished by using the SEGTYPE= parameter in the Master File as a logical key for sequential files.</td>
</tr>
<tr>
<td>FIXF</td>
<td>OFF</td>
<td>Allows you to stop the retrieval process when an equality test on this field holds true. This changes former behavior, as the interface previously read all of the records from the QSAM file and then passed them to FOCUS to apply the screening conditions when creating the final report. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON FOCUS stops the retrieval process when an equality test on this field holds true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF FOCUS does not stop the retrieval process when an equality test on this field holds true.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Options</td>
<td>Explanation</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>FOCALLOC</td>
<td>ON</td>
<td>Automatic allocation of FOCUS files in MVS only. Allocation is done based on Prefix.Master. FOCUS. The DISP will be SHR.</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>FOCUSIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOC144</td>
<td>NEW</td>
<td>Tells FOCUS to suppress warning message FOC144, which reads: “Warning Testing in Independent sets of Data.” The settings are:</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td>FOCUSIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOCSTACK</td>
<td>(a )</td>
<td>Amount of core in thousands of bytes used by FOCSTACK, the stack of FOCUS commands from FOCEXECs awaiting execution. The maximum value of FOCSTACK depends on your current region size. The default is 8. You can also specify the parameter as FOCSTACK SIZE.</td>
</tr>
<tr>
<td></td>
<td>(n)</td>
<td></td>
</tr>
<tr>
<td>HDAY</td>
<td></td>
<td>This setting is by default not set at all. Activates the holiday file that is used in conjunction with the data functions DATEDEIF, DATEMOV, DATECVT, and DATEADD.</td>
</tr>
<tr>
<td>HLISUTRACE</td>
<td>ON</td>
<td>This is a debugging setting that records the last 20 events that the FOCUS Database Server (formerly called the sink machine) performed. The information is written to memory and is intended for use when reading a dump of the SU address space. This setting may only be set in the SU profile, HLIPROF.</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td>HLISUDUMP</td>
<td>(n)</td>
<td>When set to 99999, a dump of the FOCUS Database Server address space will occur for any error on the server. The user abend code is set to 275. The user code will also be set to the error number. This setting is only used for debugging FOCUS Database Server problems and may only be set in the SU profile, HLIPROF.</td>
</tr>
<tr>
<td></td>
<td>99999</td>
<td></td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOLDATTR</td>
<td>ON</td>
<td>Controls whether the TITLE and ACCEPT attributes in the original Master File are propagated in the HOLD Master. Does not affect the way fields are named in the HOLD Master File. You can also specify the parameter as HOLDATTRS. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FOCUS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>ON</strong> Uses the TITLE attribute as specified in the original Master File in HOLD files in any format. The ACCEPT attribute, however, will be propagated to the HOLD Master only for HOLD files in FOCUS format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>OFF</strong> Does not use the TITLE or ACCEPT attributes from the original file. The HOLD Master will not contain these attributes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>FOCUS</strong> Uses the TITLE and ACCEPT attributes only for HOLD files in FOCUS format. FOCUS is the default.</td>
</tr>
<tr>
<td>HOLDLIST</td>
<td>PRINTONLY</td>
<td>Restricts fields in HOLD files to those appearing in a request. The settings are:</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>PRINTONLY</strong> Includes in the generated HOLD file only those displayable fields specified in the report request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <strong>ALL</strong> Specifies that all verb object fields and printed sort fields referenced in a request will appear in a HOLD file, including both computed fields and fields mentioned in a COMPUTE statement. ALL is the default. (OLD may be used as a synonym for ALL.)</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| HOLDSTAT  | ON\n OFF \n name | Includes comments and DBA information in HOLD Master Files. This information is derived from the IBI-supplied HOLDSTAT ERRORS file or a user-specified file. The settings are:

ON

Looks for the file HOLDSTAT MASTER or ERRORS (CMS) or the member HOLDSTAT in the PDS allocated to the ddname MASTER or the ddname ERRORS (MVS).

OFF

Ignores the HOLDSTAT file. OFF is the default.

name

Specifies a HOLDSTAT file, created by the user, to be included in the HOLD Master File. |
| HOTMENU   | ON\n OFF | Automatically displays the Hot Screen PF key legend at the bottom of the Hot Screen report. The settings are:

ON

Displays the PF key legend.

OFF

Does not display the PF key legend. To see the PF key legend, the user must press PF1. OFF is the default. |
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMLE</td>
<td>OFF ON ALL</td>
<td>The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>Does not initialize the LE environment. OFF is the default value and is the recommended setting for applications using only IBI-supplied subroutines.</td>
</tr>
<tr>
<td></td>
<td>ON</td>
<td>Establishes the LE pre-initialization environment with the IBM default configuration. This configuration initializes the LE environment for subroutines coded in COBOL, PL/I, C, and ASMH if the routines are linked with the LE environment. If the application calls a module not supported under LE, it runs without LE. ON is the recommended setting for applications that call user-written subroutines linked with the LE environment and not coded in FORTRAN. ON is also recommended for applications that call a combination of these subroutines and IBI-supplied subroutines. Running IBI-supplied subroutines with this setting requires LE version 1.5 or above.</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>Should be used only for user-created FORTRAN subroutines that need the LE environment.</td>
</tr>
</tbody>
</table>

**Note:**
- ALL is not supported by IBM. The ALL setting adds FORTRAN to the list of languages supported for LE pre-initialization. The FORTRAN run-time libraries must be installed under LE. However, FORTRAN modules do not run under LE. ALL is the recommended setting for applications that call user-written subroutines written in FORTRAN if the FORTRAN run-time libraries were installed under LE. This setting requires LE version 1.5 and above.
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
|           |         | • Non-LE Assembler subroutines require source code changes in order to be LE compliant.  
|           |         | • Mixed-mode applications calling both LE and non-LE subroutines in the same FOCEXEC or FOCUS session are not supported and may produce unpredictable results. |
| IMMEDTYPE | ON      | Used with TOE, tells FOCUS where to send line mode output. The settings are: |
|           | OFF     | ON  
|           |         | Sends all line mode output, such as -TYPE, to the Output Window as it is executed, line by line.  
|           |         | OFF  
|           |         | Buffers all line mode output. The output appears in the Output Window as a new full screen. OFF is the default. |
| IMS       | NEW     | Tells FOCUS whether to use the new version of the IMS interface. The new version is for releases after 6.8 PUT level 9406. The settings are: |
|           | OLD     | NEW  
|           |         | Is the new version of the IMS interface. NEW is the default.  
|           |         | OLD  
|           |         | Is the version of IMS interface prior to release 6.8 PUT level 9406. |
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>NEW</td>
<td>The indexing scheme used for indexes (fields specified with FIELDTYPE=I keywords in the Master Files). You can also specify the parameter as INDEX TYPE. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OLD</td>
</tr>
<tr>
<td>JOINOPT</td>
<td>GNTINT</td>
<td>Allows the joining of two files that contain different numeric data types.</td>
</tr>
<tr>
<td>P1170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANG</td>
<td>ID name abbrev</td>
<td>Specifies the National Language Support (NLS) environment which allows for multiple language ERRORS files. You can also specify the parameter as LANGUAGE. The settings are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ENGLISH</td>
<td>AME</td>
</tr>
<tr>
<td>1</td>
<td>AMENGLISH</td>
<td>AME</td>
</tr>
<tr>
<td>20</td>
<td>ARABIC</td>
<td>ARB</td>
</tr>
<tr>
<td>359</td>
<td>BULGARIAN</td>
<td>BLG</td>
</tr>
<tr>
<td>416</td>
<td>CANFRENCH</td>
<td>CFR</td>
</tr>
<tr>
<td>34</td>
<td>CATALAN</td>
<td>CAT</td>
</tr>
<tr>
<td>85</td>
<td>S–CHINESE</td>
<td>PRC</td>
</tr>
<tr>
<td>86</td>
<td>T–CHINESE</td>
<td>ROC</td>
</tr>
<tr>
<td>45</td>
<td>DANISH</td>
<td>DAN</td>
</tr>
<tr>
<td>31</td>
<td>DUTCH</td>
<td>DUT</td>
</tr>
<tr>
<td>358</td>
<td>FINNISH</td>
<td>FIN</td>
</tr>
<tr>
<td>32</td>
<td>FLEMISH</td>
<td>FLM</td>
</tr>
</tbody>
</table>
### Parameter Options Explanation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG (continued)</td>
<td>ID name abbrev</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>FRENCH FRE</td>
</tr>
<tr>
<td></td>
<td>49</td>
<td>GERMAN GER</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>GREEK GRE</td>
</tr>
<tr>
<td></td>
<td>972</td>
<td>HEBREW HEB</td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>HINDI IND</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>HUNGARIAN HUN</td>
</tr>
<tr>
<td></td>
<td>354</td>
<td>ICELANDIC ICL</td>
</tr>
<tr>
<td></td>
<td>39</td>
<td>ITALIAN ITA</td>
</tr>
<tr>
<td></td>
<td>81</td>
<td>JAPANESE JPN</td>
</tr>
<tr>
<td></td>
<td>10081</td>
<td>JAPANESE-E JPN</td>
</tr>
<tr>
<td></td>
<td>82</td>
<td>KOREAN KOR</td>
</tr>
<tr>
<td></td>
<td>47</td>
<td>NORWEGIAN NOR</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>POLISH POL</td>
</tr>
<tr>
<td></td>
<td>351</td>
<td>PORTUGUESE POR</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>RUSSIAN RUS</td>
</tr>
<tr>
<td></td>
<td>38</td>
<td>SLOVENIAN SLO</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>SPANISH SPA</td>
</tr>
<tr>
<td></td>
<td>46</td>
<td>SWEDISH SWE</td>
</tr>
<tr>
<td></td>
<td>66</td>
<td>THAI THA</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>TURKISH TUR</td>
</tr>
<tr>
<td></td>
<td>44</td>
<td>UKENGLISH UKE</td>
</tr>
</tbody>
</table>
Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>LANG</td>
<td>ID name abbrev</td>
<td>To specify JAPANESE-E, you can use the ID or the full name, but not the abbreviation JPN. In addition, when you select JAPANESE-E. Make sure that the TERM parameter is set to a value that supports DBCS. The syntax is</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SET TERM =</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{IBM5550}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{F6650}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{H56020}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>where:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM5550</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies an IBM 5550 or a PS/55 terminal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F6650</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies a Facom F-6650 terminal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>H56020</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies a Hitachi H-560/20 terminal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These terminals support DBCS. The default value for TERM is IBM3270, which does not support DBCS.</td>
</tr>
<tr>
<td>LEADZERO</td>
<td>ON OFF</td>
<td>Leading zeros are truncated in Dialogue Manager strings. The subroutines in FOCUS, when called in Dialogue Manager, may return a numeric result. If the format of the result is YMD and contains a 00 for the year, the 00 is truncated. The settings are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allows the display of leading zeros if they are present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Truncates leading zeros if they are present.</td>
</tr>
<tr>
<td>LEFTMARGIN</td>
<td>value</td>
<td>Sets the left boundary of report contents on a page. The default value is .25 inches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This command applies only to PostScript and PDF report formats.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Options</td>
<td>Explanation</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>LINES</td>
<td>( n )</td>
<td>Number of lines per page of printed output, from the page heading at the top to the footing on the bottom. If this value is less than the value set for PAPER, the difference provides a bottom margin. FOCUS never puts more lines on a page than the LINES parameter specifies, but may put less. The value of LINES can range between 1 and 999999 (use the value 999999 for continuous forms). Note that when the ? SET query lists this parameter, it labels the value LINES/PRINT. The default is 57. <strong>Note:</strong>&lt;br&gt;• When the STYLESHEET parameter is in effect, the setting for LINES is ignored.&lt;br&gt;• When using SKIP-LINE in a report always set LINES to at least one less than PAPER to avoid unintentional page breaks at the bottom of the page.</td>
</tr>
<tr>
<td>MAXLRECL</td>
<td>( n )</td>
<td>Defines the maximum record length for an external file with OCCURS segments. The default is 0. However, FOCUS can read a 16K re cl by default. This may be set to a maximum of 32K.</td>
</tr>
<tr>
<td>MASTER</td>
<td>NEW  OLD</td>
<td>Used with the FUSION option. New Master Files passes for blank delimited Master Files, which use the new FUSION syntax.</td>
</tr>
<tr>
<td>MESSAGE MSG</td>
<td>ON OFF</td>
<td>Select display of FOCUS informational messages.&lt;br&gt;<strong>ON</strong>&lt;br&gt;Displays informational messages. ON is the default.&lt;br&gt;<strong>OFF</strong>&lt;br&gt;Suppresses informational messages. OFF also suppresses the carets that appear when FOCUS executes commands in FOCEXECs. Error messages and carets that prompt for input are not suppressed.</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| MINIO     | ON, OFF | A buffering technique that improves performance by reducing I/O operations up to fifty percent when accessing FOCUS data sources under MVS. The settings are:  
  - **ON**: Enables MINIO. This setting is the default.  
  - **OFF**: Disables MINIO.  
When MINIO is ON, no block is ever read more than once, and therefore the number of reads performed will be the same as the number of tracks present. This results in an overall reduction in elapsed times when reading and writing.  
With FOCUS data sources that are not disorganized, MINIO can greatly reduce the number of I/O operations for TABLE and MODIFY commands. The actual I/O reduction will vary depending on data source structure and average number of children segments per parent segment. By reducing I/O operations, elapsed time for TABLE and MODIFY commands also drop. |
| NODATA | *string* | Character string used to indicate missing data in reports. The default is a period (.). |
| OFFLINE-FMT | | Reserved for future use with StyleSheets. |
### Parameter Options Explanation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLINE-FMT</td>
<td>STANDARD</td>
<td>StyleSheet reports are generated in PostScript format. Styled reports can only be printed on a PostScript printer. The settings are:</td>
</tr>
<tr>
<td></td>
<td>POSTSCRIPT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PS</td>
<td></td>
</tr>
<tr>
<td>ORIENTATION</td>
<td>PORTRAIT</td>
<td>Used with StyleSheets, determines the orientation of your report page. The settings are:</td>
</tr>
<tr>
<td></td>
<td>LANDSCAPE</td>
<td></td>
</tr>
<tr>
<td>PAGE[-NUM]</td>
<td>ON</td>
<td>Select page numbering of FOCUS output. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOPAGE</td>
<td></td>
</tr>
</tbody>
</table>

#### ONLINE-FMT

- **STANDARD**
  - Is the default.
- **POSTSCRIPT**
  - The report output goes to a PostScript file with the name PSOUT. In MVS, the PostScript formatted report output is in a variable length PDS allocated to the ddname PS. In CMS, the output is in a file with the file type PS. The parameters set with the SET STYLESHEET command are in effect.
- **PS**
  - Is a synonym for POSTSCRIPT.

#### ORIENTATION

- **PORTRAIT**
  - Specifies a report page that is longer than it is wide. PORTRAIT is the default.
- **LANDSCAPE**
  - Specifies a report page that is wider than it is long.

#### PAGE[-NUM]

- **ON**
  - Numbers the pages and displays the page number on the upper left-hand corner of the page. ON is the default.
- **OFF**
  - Suppresses page numbering.
- **NOPAGE**
  - Suppresses page numbers and page breaks, causing the report to be printed as one long page. Note that the LINES parameter controls where column headings are printed.
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PAGESIZE</strong></td>
<td>size</td>
<td>Used with StyleSheets, selects the paper size for report pages. For valid SIZE settings, refer to the <em>Creating Reports</em> manual. For optimal report appearance, the actual paper size must match your setting for PAGESIZE or your report or your report will be cropped or contain extra blank spaces.</td>
</tr>
<tr>
<td><strong>PANEL</strong></td>
<td>( \leq n )</td>
<td>The maximum line width (characters) of a report panel. If a FOCUS report exceeds this value, FOCUS partitions each page of output into sections called panels. For example, if you set PANEL to 80, then the first 80 characters of each record appear on the first panel, the second 80 characters appear on the second panel, and so on. FOCUS can divide a report into a maximum of four panels when SET BYPANEL is set to OFF. The report may be divided into 99 panels if SET BYPANEL is specified. If PANEL is 0, FOCUS does not divide the report into panels. The default is 0. The best setting for PANEL is the screen width of your terminal (usually 80) or the print width of your printer (usually 132). If the PANEL setting is larger (or 0), the long report lines wrap around the screen or page. <strong>Note:</strong> When the STYLESHEET parameter is in effect, the setting for PANEL is ignored.</td>
</tr>
</tbody>
</table>
**Parameter** | **Options** | **Explanation**
--- | --- | ---
PAPER LINES/PAGE | \[66 \quad n\] | Physical length of the paper used for output. This is the length of a sheet in inches multiplied by the number of lines printed per inch. If your printer prints six lines per inch on standard 11 in. forms, this setting should be 66, unless you are placing a footing at the bottom of the page. In that case, set PAPER to 62. The value of PAPER can range from 1 to 999999 (use the value 999999 for continuous forms). The default is 66. **Note:**
- When the STYLESHEET parameters is in effect, the setting for PAPER is ignored.
- When the PSET command lists this parameter it labels the value LINES/PAGE.

PASS | password \[\text{IN filename}\] | Gives you access to FOCUS files or a specific data source (file name) that is protected by FOCUS database security.

PAUSE | ON \[\text{OFF}\] | Selects a pause before displaying FOCUS reports on the terminal. When you use a printing terminal, this parameter allows you to adjust the paper before printing the report. **ON**

Activates pausing. ON is the default. **OFF**

Deactivates pausing.

When the SCREEN parameter is ON, the PAUSE parameter is set ON (until you set the PAUSE parameter to OFF). If you set the SCREEN parameter OFF, the PAUSE parameter is set OFF. Note that you can change the PAUSE parameter without affecting the SCREEN parameter.

Note that this setting does not affect offline printing (routing output to a system printer).
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PFnn</strong></td>
<td>function</td>
<td>Assigns a function to the PF key specified by ( nn ), enabling you to change the current PF key setting when using FIDEL (and also, under certain conditions, within the Window facility). The current settings are displayed by the <code>?PFKEY</code> command.</td>
</tr>
<tr>
<td><strong>PREFIX</strong></td>
<td>prefix</td>
<td>For MVS only, specifies the prefix of existing data sets automatically allocated by FOCUS. The default setting in TSO is your userid; the default setting in batch is FOCUS.</td>
</tr>
</tbody>
</table>
| **PRINT** | ONLINE, OFFLINE | Report output destination. The settings are:  
- **ONLINE**  
  The terminal. ONLINE is the default.  
- **OFFLINE**  
  The system printer. To specify a particular printer.  

Note that you can enter ONLINE and OFFLINE as separate commands which have the same effect as using ONLINE and OFFLINE as PRINT settings. |
Developing Applications 4-31

Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINTPLUS</td>
<td>ON, OFF</td>
<td>Introduces enhancements to the display alternatives offered by the FOCUS Report Writer. The settings are ON and OFF. With PRINTPLUS ON, the PAGE-BREAK is handled internally to provide the correct spacing of pages, NOSPLIT is handled internally and you can perform RECAPs in cases where pre-specified conditions are met. Additionally, a Report SUBFOOT now prints above the footing instead of below it. ON is the default.</td>
</tr>
</tbody>
</table>

Note:

- To force a break at a specific spot, you must use NOSPLIT. 
- PRINTPLUS is not supported with StyleSheets. A warning message is generated in this case. 
- Problems may be encountered if HOTSCREEN is set OFFLINE. A warning message is generated.
### Parameter: QUALCHAR

<table>
<thead>
<tr>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| character | Specifies the qualifying character to be used in qualified field names. Valid qualifying characters are:  
  - period (hex 4B)  
  - colon (hex 7A)  
  - exclamation point (hex 5A)  
  - percent sign (hex 6C)  
  - broken vertical bar (hex 6A)  
  - backslash (hex E0)  

Apart from the period, the use of all the other qualifying characters listed above is restricted, and should not be used with 66-character field names.

If the qualifying character is a period, you can use any of the other characters listed above as part of a field name. On the other hand, if you change the default qualifying character to a character other than the period, then you cannot use that character in a field name.

In VM, if the TERM tabchar is ON or if the CMS INPUT command includes the broken vertical bar (hex 6A), then the broken vertical bar cannot be the qualifying character. To query INPUT, type Q INPUT at the CMS prompt. |

### Parameter: QUALTITLES

<table>
<thead>
<tr>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| ON | Determines whether duplicate field names appear as qualified column titles in reports. The settings are:  
  - ON Enables qualified column titles when duplicate field names exist and FIELDNAME=NEW.  
  - OFF Disables qualified column titles. OFF is the default. |

### Parameter: REBUILDMSG

<table>
<thead>
<tr>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>Allows for direct control over the frequency with which REBUILD issues messages.</td>
</tr>
</tbody>
</table>
### Developing Applications

#### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RECAP-COUNT</strong></td>
<td>ON</td>
<td>Tells FOCUS when recap lines are to be counted even though they are not displayed when SUBFOOT is used. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indicates that recap lines are to be counted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indicates that recap lines are not to be counted. OFF is the default.</td>
</tr>
<tr>
<td><strong>RECORDLIMIT</strong></td>
<td></td>
<td>Tells FOCUS to limit the number of records retrieved.</td>
</tr>
<tr>
<td><strong>RIGHTMARGIN</strong></td>
<td>value</td>
<td>Sets the right boundary of report contents on a page. The default value is .25 inches.</td>
</tr>
<tr>
<td><strong>RPAGESET</strong></td>
<td>NEW</td>
<td>Tells FOCUS how to control requests combining RECAP with SUBFOOT. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OLD</td>
<td>NEW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sets lines per page equal to LINES+2+the number of the highest BY field with a SUBFOOT.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Works as in release 6.0. OLD is the default.</td>
</tr>
<tr>
<td><strong>SAVEMATRIX</strong></td>
<td>ON</td>
<td>Preserves the internal matrix and keeps it available for subsequent RETYPE, HOLD, SAVE, SAVB, and REPLOT commands when followed by Dialog Manager commands. The settings are:</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enables the SAVEMATRIX facility and causes FOCUS to save the last internal matrix generated. ON is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Does not guarantee that the internal matrix will be available.</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| SBORDER   | ON, OFF | Generates a solid border on the screen for full-screen mode. The settings are:  
ON
  Enables solid borders. ON is the default.  
OFF
  Enables dashed (nonsolid) borders.  
If the screen appears to be generated incorrectly, it is possible that the terminal does not support this new feature; change the setting to OFF to correct the situation.  
The amper variable &FOCSBORDER contains the value of the SBORDER setting. &FOCSBORDER may be included in the Dialogue Manager -TYPE command. |
| SCREEN    | ON, OFF, PAPER | Selects the Hot Screen facility.  
ON
  Activates the Hot Screen facility. ON is the default.  
OFF
  Deactivates the Hot Screen facility.  
PAPER
  Activates the Hot Screen facility and causes FOCUS to use the settings for LINES and PAPER parameters to format screen display.  
When the SCREEN parameter is ON, the PAUSE parameter is set ON (until you set the PAUSE parameter OFF). If you set the SCREEN parameter OFF, the PAUSE parameter is set OFF. Note that you can change the PAUSE parameter without affecting the SCREEN parameter. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHADOW</td>
<td>ON, OFF, OLD</td>
<td>Activates the Absolute File Integrity feature. You can also specify the parameter as SHADOW PAGE. The settings are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON: Activates the Absolute File Integrity feature. The maximum number of pages shadowed is 256K.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF: Deactivates the Absolute File Integrity feature. OFF is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OLD: Indicates that your FOCUS file was created before Release 7.0. This means that the maximum number of pages shadowed is 63,551.</td>
</tr>
<tr>
<td>SHIFT</td>
<td>ON, OFF</td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON: Specifies a shift string for Hebrew or DBCS (double byte character support).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OFF: Indicates that SHIFT is not in effect. OFF is the default.</td>
</tr>
<tr>
<td>SORTLIB</td>
<td>VMSORT, SYNCSORT, DFSORT, SITEDEFINED</td>
<td>For FOCUS VM/CMS. Used to tell FOCUS which sort package is installed at your site. Use SITEDEFINED if not VMSORT, SYNCSORT, or DFSORT. This sort package must be installed in SORTLIB TXTLIB in order for FOCUS to find it.</td>
</tr>
</tbody>
</table>
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| **SPACES** | AUTO \(n\) | Controls the number of spaces between columns in a report. The settings are:  

- **AUTO**  
  Specifies that FOCUS should put either one or two spaces between columns. AUTO is the default.  
- **\(n\)**  
  Is a number from 1 to 8. |
| **SQLTOPTTF** | ON \(\text{OFF}\) | Enables the SQL Translator to generate TABLEF commands instead of TABLE commands.  

- **ON**  
  Causes TABLEF commands to be generated when possible (for example, if there is no JOIN or GROUP BY phrase).  
- **OFF**  
  Causes TABLE commands to be generated. OFF is the default. |
| **SQUEEZE** | ON \(\text{OFF}\) | Determines how StyleSheets will assign column widths in a report.  

- **ON**  
  Assigns column widths based on the widest data value or widest column title, whichever is longer. This is the default.  
- **OFF**  
  Assigns column widths based on the field format specified in the file description. OFF will pad the column widths up to the length of the column title or field format descriptions, whichever is greater. |
### Parameter Options Explanation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STYLE</strong></td>
<td><strong>ON</strong></td>
<td>Controls report formatting. Report pages can be defined by style parameters that specify characteristics such as page size, orientation, units, and margins. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td></td>
</tr>
<tr>
<td><strong>stylesheet</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>stylesheetname</strong></td>
<td></td>
<td>Is the 8-character name of a StyleSheet file. This file has a file type of FOCSTYLE in CMS; in MVS or MVS, it is a member of the PDS allocated to the name FOCSTYLE.</td>
</tr>
<tr>
<td><strong>SUMPREFIX</strong></td>
<td><strong>FST</strong></td>
<td>When an external sort product performs aggregation of alphanumeric or smart date formats, the order of the answer set returned differs from the order of the FOCUS sorted answer sets. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>LST</strong></td>
<td></td>
</tr>
<tr>
<td><strong>susi</strong></td>
<td></td>
<td>See the <em>Simultaneous Usage Reference Manual for TSO</em>.</td>
</tr>
</tbody>
</table>
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUTABSIZE</td>
<td></td>
<td>See the <em>Simultaneous Usage Reference Manual for TSO</em>.</td>
</tr>
<tr>
<td>TEMP[DISK]</td>
<td>disk</td>
<td>For CMS only, determines the disk FOCUS uses for temporary work space, and to store extract files (HOLD and SAVE).</td>
</tr>
<tr>
<td>TERM</td>
<td>terminal type</td>
<td>Selects the terminal type. You can also specify the parameter as TERMINAL. The default is IBM3270.</td>
</tr>
</tbody>
</table>
| TESTDATE    | yyyyymmd or TODAY | Allows you to temporarily alter your FOCUS system date for a given application program. This allows you to determine what impact the year 2000 will have on your application programs.  

The settings for TESTDATE are:

- **yyyyymmd**: Is an 8-digit date in the format YYYYMMDD.
- **TODAY**: Is today’s date. TODAY is the default.

**Note:**

- Only use TESTDATE for testing purposes with test data.
- The value of TESTDATE affects all reserved variables that retrieve the current date from the system. Setting TESTDATE also affects anywhere in FOCUS that a date is used (such as CREATE, MODIFY, MAINTAIN) but does not affect the date referenced directly from the system.
- TESTDATE can either be equal to TODAY or a date in the format YYYYMMDD. If anything else is entered the following message is displayed:

```
TESTDATE MUST BE YYYYMMDD OR TODAY
```
FOCUS text fields have been enhanced significantly with Release 7.0, and a new SET command introduced to preserve downward compatibility with prior FOCUS releases. The settings are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXTFIELD</td>
<td>OLD</td>
<td>Enables you to use text field data in prior releases of FOCUS when that data has been created or modified in Release 7.0. OLD is the default.</td>
</tr>
<tr>
<td>TXTFIELD</td>
<td>NEW</td>
<td>Disables the ability to use text field data in prior FOCUS releases when that data has been created or modified in Release 7.0.</td>
</tr>
<tr>
<td>TITLES</td>
<td>ON</td>
<td>Selects pre-defined column titles from the Master File. ON is the default.</td>
</tr>
<tr>
<td>TITLE</td>
<td>OFF</td>
<td>Selects the field names in the Master File to use as report column titles.</td>
</tr>
<tr>
<td>TOPMARGIN</td>
<td>value</td>
<td>Sets the top boundary of report contents on a page. The default value is .25 inches.</td>
</tr>
</tbody>
</table>

**Note:** FOCUS Release 7.0 preserves text fields exactly as they are entered into a data source with the ON MATCH/NOMATCH TED command. See the *Overview and Operating Environment* manual for additional information.
### Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACKIO</td>
<td><strong>ON</strong></td>
<td>MVS FOCUS gathers more pages to fill a track before reading or writing the pages to disk. This results in significant reductions in I/O requirements and in elapsed time for FOCUS files. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong> Enables FOCUS to fill a track before reading or writing to disk. ON is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Disables TRACKIO.</td>
</tr>
<tr>
<td>TRMOUT</td>
<td><strong>ON</strong></td>
<td>Suppresses all output messages to the terminal. The settings are:</td>
</tr>
<tr>
<td></td>
<td><strong>OFF</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ON</strong> Displays output messages to the terminal. ON is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>OFF</strong> Suppresses messages to the terminal.</td>
</tr>
<tr>
<td>TRMSW</td>
<td><strong>80</strong></td>
<td>Terminal screen width is 80 characters. Cannot be set.</td>
</tr>
<tr>
<td>TRMSD</td>
<td><strong>24</strong></td>
<td>Terminal screen depth is 24 characters and can’t be set otherwise.</td>
</tr>
<tr>
<td>TRMTYP</td>
<td><strong>1 (3270)</strong></td>
<td>Terminal type. Cannot be set.</td>
</tr>
</tbody>
</table>
## Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| UNITS       | **INCHES**  
*CM*  
**PTS** | Sets the measurement units that FOCUS uses to calculate margins.  
**Note:** This command applies only to PostScript and PDF report formats.  
The settings are:  
**INCHES**  
Specifies that margin values are measured in inches.  
**CM**  
Specifies that margin values are measured in centimeters.  
**PTS**  
Specifies that margin values are measured in points. (one inch = 72 points, one cm = 28.35 points) |
| USER        | password | Allows you access to FOCUS files that are protected by FOCUS database security. |
| WIDTH       | **130**  
*n* | Value indicates the logical record length of your output data set. The default is 130.  
**Note:** When the STYLESHEET parameter is in effect, FOCUS ignores the setting for WIDTH. |
| XRETRIEVAL  | **ON**  
**OFF** | Previews the format of a report without actually accessing any data. This parameter enables you to perform TABLE, TABLEF, or MATCH requests and produce HOLD Master Files without processing the report. The settings are:  
**ON**  
Specifies retrieval is to be performed. ON is the default.  
**OFF**  
Specifies that no retrieval is to be performed. |
Setting Parameters: SET

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Options</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>YRTHRESH</td>
<td></td>
<td>Sets YRTHRESH as an offset from the current year in addition to specifying a year. This technique creates a moving century window that increments itself each year without modifying your application. You decide the number of years to offset in YRTHRESH. For example, if the current year is 1998 and you wish to set YRTHRESH to 60, you can set YRTHRESH to -38 (1998 - 38 = 60). By setting YRTHRESH to a negative number FOCUS subtracts, in this example, 38 from whatever the current year is. In the year 1999 YRTHRESH is 61 instead of 60 (1999 - 38 = 61) illustrating how the moving window application functions without outside intervention. If you set YRTHRESH to a large enough value that crosses a century boundary, the value of DEFCENT is recalculated. For example, if you set YRTHRESH to minus 99 (1998 - 99 = -1), DEFCENT is calculated to 18 and YRTHRESH becomes 99. The 100-year span begins with a pivot year of 1899 and ends with year 1998. ? SET and ? SET ALL now reflect the new settings of DEFCENT.</td>
</tr>
<tr>
<td>YRT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5 Defining Word Substitutions: LET

Topics:
- LET Command Syntax
- Variable Substitutions in LET Phrases
- Checking Current LET Substitutions: ? LET
- Clearing LET Substitutions: LET CLEAR
- Saving LET Equivalences in a File: LET SAVE

The LET command enables you to define words to represent other words and phrases. When you enter LET-defined words, FOCUS translates the words into the phrases they represent. The result is the same as if you entered the phrases directly.

When you substitute words for phrases, you create your own FOCUS statements and commands. This reduces the typing necessary to issue requests, especially when using phrases repeatedly, and makes your requests easier to understand.

You can substitute any phrase that you enter online unless you are entering a MODIFY request.

You cannot substitute phrases used in Dialogue Manager control statements. For example, if you substituted the word YES for the phrase EQ Y, the following would be invalid:

\[-IF \&\textit{ANSWER} YES \textit{GOTO CONT}\]

Once executed, the LET substitutions last until they are cleared or until the FOCUS session ends. To use the same substitutions in many sessions, place the LET commands in a stored procedure, which you can then execute with one short command. If you want to save LET statements currently active, use the LET SAVE facility described in Saving LET Equivalences in a File: LET SAVE, on page 5-12.

There are several utilities used with the LET command. They are:
- The ? LET query to list currently defined LET substitutions.
- The LET CLEAR command to clear LET substitutions.
- The LET SAVE command to save LET substitutions in a FOCEXEC file.
- The LET ECHO facility to display a FOCUS statement as interpreted by FOCUS.
Defining Word Substitutions: LET

**Example** Defining Word Substitutions

For example, if you define the word WORKREPORT to mean TABLE FILE EMPLOYEE, you issue:

```
WORKREPORT
PRINT LAST_NAME
END
```

FOCUS generates the following report request:

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME
END
```

**Example** Defining Substitutions for Translation

Non-English speakers can use LET commands to translate FOCUS statements into another language. For example, this stored procedure

```
TABLE FILE CAR
SUM AVE.RCOST OVER AVE.DCOST
BY CAR ACROSS COUNTRY
END
```

could be translated into French as:

```
CHARGER FICHIER CAR
SOMMER AVE.RCOST SUR AVE.DCOST
PAR CAR TRAVERS COUNTRY
END
```
LET Command Syntax

The LET command has a short form and a long form. Use the short form for one or two LET statements that fit on one line. Otherwise, use the long form.

Syntax

**LET Command: The Short Form**

The syntax of the short form of the LET command is

\[
\text{LET word} = \text{phrase} \; [; \text{word} = \text{phrase}...]
\]

where:

- **word**
  Is a character string of up to 80 characters with no embedded blanks.

- **phrase**
  Is a character string of up to 256 characters including embedded blanks. The phrase can also include other special characters, but semicolons (;) and pound signs (#) need special consideration.

The following example defines the word **WORKREPORT** as a substitute for the phrase **FILE EMPLOYEE**:

\[
\text{LET WORKREPORT} = \text{TABLE FILE EMPLOYEE}
\]

If the word you are defining appears in the phrase, you must enclose it in single quotation marks:

\[
\text{LET TABLE} = \text{'TABLE'} \text{ FILE EMPLOYEE}
\]

To define several words in one LET command, separate the statements with semicolons (;). For example:

\[
\text{LET WORKREPORT=}\text{TABLE FILE EMPLOYEE;} \; \text{PR=}\text{PRINT}
\]

This LET command substitutes **WORKREPORT** for the phrase **TABLE FILE EMPLOYEE**, and **PR** for the word **PRINT**; the two statements are separated by a semicolon. Remember, the short form of the LET command cannot exceed one line.
Defining Word Substitutions: LET

**Syntax**

**LET Command: The Long Form**

Use the long form of the LET command for definitions that exceed one line. The syntax is

```
LET
 word = phrase
 .
 .
 .
END
```

**where:**

- `word` is a character string of up to 80 characters with no embedded blanks.
- `phrase` is a character string of up to 256 characters including embedded blanks. The keywords `LET` and `END` must be entered on separate lines. For example:

```
LET
 RIGHTNAME = 'STEVENS' OR 'SMITH' OR 'JONES' OR 'BANNING' OR 'MCCOY' OR 'MCKNIGHT'
END
```

As with the short form, you can define several words in one LET command, separating definitions with semicolons (;). A definition starting on a new line does not need a semicolon after the preceding statement.

LETs are terminated by end-of-line characters (;) followed by another LET declaration or END.

```
LET
 WORKREPORT = TABLE FILE EMPLOYEE; PR = PRINT
 RIGHTNAME = 'STEVEN' OR 'SMITH' OR 'JONES'
END
```
Variable Substitutions in LET Phrases

Using the LET command, you can define words that represent variable phrases. Variable phrases contain placeholder symbols < > to indicate missing elements in the phrase. This allows you to give phrases different meanings in different FOCUS statements.

For example, this LET statement

\texttt{LET UNDERSCORE = ON < > UNDER-LINE}

contains one placeholder. After entering this LET statement, you can use the word UNDERSCORE as shown in this FOCUS request:

\begin{verbatim}
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY EMP_ID BY HIRE_DATE
UNDERSCORE EMP_ID
END
\end{verbatim}

In the statement UNDERSCORE EMP_ID, the word EMP_ID follows the defined word UNDERSCORE. FOCUS inserts this word into the placeholder and translates UNDERSCORE EMP_ID into:

\begin{verbatim}
ON EMP_ID UNDER-LINE
\end{verbatim}

The part of the statement following the LET-defined word supplies the missing characters to the placeholders.

Placeholders can be unnumbered or numbered. If the placeholders are not numbered, then they are filled from left to right: the first word in the FOCUS statement after the LET-defined word fills the first placeholder, the second word fills the second placeholder, and so on to the last placeholder. For example, if you enter

\begin{verbatim}
LET TESTNAME = WHERE LAST_NAME IS < > OR < >
\end{verbatim}

and then enter the statement in your request

\begin{verbatim}
TESTNAME 'MCKNIGHT' 'STEVENS' 'BLACKWOOD'
\end{verbatim}

FOCUS translates this statement into:

\begin{verbatim}
WHERE LAST_NAME IS 'MCKNIGHT' OR 'STEVENS' OR 'BLACKWOOD'
\end{verbatim}

Notice that the variable phrase needs no placeholder at the end. Once all the placeholders are filled, the rest of the FOCUS statement just follows naturally. In this example, the words MCKNIGHT and STEVENS filled the placeholders. BLACKWOOD was left over, so it follows after the variable phrase.

You can number the placeholders:

\begin{verbatim}
LET TESTNAME = WHERE LAST_NAME IS <1> OR <2> OR <3>
\end{verbatim}
Defining Word Substitutions: LET

FOCUS then fills the placeholders in numerical order. If two placeholders have the same number, FOCUS fills both with the same word. For example, if you enter this LET statement

LET RANGE = SUM MAX.<1> AND MIN.<1>

and then enter this statement

RANGE SALARY

FOCUS translates the statement as:

SUM MAX.SALARY AND MIN.SALARY

If you do not supply enough words to fill in all the placeholders, the extra placeholders are null. For example, if you enter this LET statement

LET TESTNAME = WHERE LAST_NAME IS < > OR < > OR

and then enter this statement

TESTNAME 'MCCOY'

FOCUS translates this statement into:

WHERE LAST_NAME IS 'MCCOY' OR OR

This statement is illegal and produces an error message.

Placeholders can be parts of words within phrases. For example, if you enter this LET statement

LET BIGGEST = MAX.<>

and later enter this statement

WRITE BIGGEST SALARY

FOCUS translates the statement as:

WRITE MAX.SALARY

Variable substitutions are useful for representing system commands. For example, in MVS:

LET ALFOC = TSO ALLOC F(<>) DA(<).FOCUS) SHR
LET LISTMEM = TSO LISTDS < > MEMBERS
Using LET Phrases in COMPUTE and DEFINE Statements

COMPUTE and DEFINE statements must have semicolons as terminators. To substitute such statements, add a second semicolon. For example:

```
LET
SALTEST = LEVEL/A4 = IF SALARY GT 35000 THEN HIGH
ELSE LOW;;
END
```

Note the double semicolon after the word LOW. When you issue the statement

```
AND COMPUTE SALTEST
```

FOCUS translates the statement as

```
AND COMPUTE LEVEL/A4 = IF SALARY GT 35000 THEN HIGH
ELSE LOW;
```

with one semicolon after the word LOW. Note that you cannot substitute phrases containing embedded semicolons.

Null Substitutions

You can define a null word (a word that is ignored by FOCUS). To define a null word, enter:

```
LET word=;
```

This allows you to replace one phrase with several LET-defined words, making your requests more readable. For example, you enter this LET command:

```
LET
DISPLAY=;
AVESAL = SUM AVE.SALARY BY DEPARTMENT
END
```

Later, you issue this request:

```
TABLE FILE EMPLOYEE
DISPLAY AVESAL
WHERE DEPARTMENT IS 'PRODUCTION'
END
```

FOCUS translates the request into:

```
TABLE FILE EMPLOYEE
SUM AVE.SALARY BY DEPARTMENT
WHERE DEPARTMENT IS 'PRODUCTION'
END
```
Defining Word Substitutions: LET

### Multiple-Line Substitutions

Many FOCUS statements, such as the END and MODIFY statements, must appear on separate lines. To substitute these phrases, place a number sign # and a space before the phrase to indicate a new line. This technique allows you to substitute one word for several lines of text. For example, you enter this LET command:

```plaintext
LET HOLDREP = ON TABLE HOLD # END
```

Later, you enter this TABLE request:

```plaintext
TABLE FILE EMPLOYEE
SUM AVE.GROSS BY EMP_ID BY PAY_DATE HOLDREP
```

FOCUS translates the request into:

```plaintext
TABLE FILE EMPLOYEE
SUM AVE.GROSS BY EMP_ID BY PAY_DATE ON TABLE HOLD
```

Special considerations regarding number signs apply in the CMS environment.

### Recursive Substitutions

A phrase in one LET statement can contain a word defined in another LET statement. For example:

```plaintext
LET
TESTNAME=IF LAST_NAME IS RIGHTNAME
RIGHTNAME = STEVENS OR MCKNIGHT OR MCCOY
END
```

The word RIGHTNAME in the phrase in the first statement is defined by the second statement. (Note that the two phrases in the LET statement could be reversed.) This LET command is the equivalent of:

```plaintext
LET
TESTNAME = IF LAST_NAME IS STEVENS OR SMITH OR MCCOY
END
```

This is called recursive substitution.

You can use recursive substitution to abbreviate long phrases within LET commands. For example:

```plaintext
LET
TESTNAME = STEVENS OR SMITH OR MCCOY OR CONT1
CONT1 = BANNING OR IRVING OR ROMANS OR CONT2
CONT2 = JONES OR BLACKWOOD
END
```
Variable Substitutions in LET Phrases

You can then use TESTNAME in this TABLE request:

```
TABLE FILE EMPLOYEE
PRINT SALARY BY LAST_NAME
IF LAST_NAME IS TESTNAME
END
```

This is the equivalent of:

```
TABLE FILE EMPLOYEE
PRINT SALARY BY LAST_NAME
IF LAST_NAME IS STEVENS OR SMITH OR MCOY OR
   BANNING OR IRVING OR ROMANS
   OR JONES OR BLACKWOOD
END
```

### Checking Current LET Substitutions: ? LET

The ? LET command displays the currently active LET substitutions.

#### Syntax

**? LET Command**

The syntax is

```plaintext
? LET [word1  word2  ... wordn]
```

where `word1` through `wordn` are the LET-defined words you want displayed. If you omit the parameters, ? LET displays a list of all active LET equivalences. The left column contains the LET-defined words; the right column contains the phrases the words represent. For example, entering

```
? LET CHART TESTNAME RIGHTNAME
```

Note that the ? LET command displays the phrase just as you entered it in the LET statement and does not make any substitutions.

To display a list of all LET substitutions in effect, issue the following command from a stored procedure:

```
? LET
```

A sample list appears like:

```
>
PR PRINT
RANGE SUM MAX.<1> AND MIN.<1>
TABLE 'TABLE' FILE EMPLOYEE
AVERAGE AVE.< >
RIGHTNAME 'STEVENS' OR 'SMITH' OR 'JONES'
UNDERSCORE ON < > UNDER-LINE
WORKREPORT TABLE FILE EMPLOYEE
```
Defining Word Substitutions: LET

Interactive LET Query: LET ECHO

The LET ECHO facility shows how FOCUS interprets FOCUS statements. This facility is a diagnostic tool you can use when statements containing LET-defined words are not being interpreted the way you expect them to. Enter:

```
LET ECHO
```

This turns on the LET ECHO facility. When you enter a FOCUS statement, LET ECHO displays the statement as interpreted by FOCUS.

**Note:**
- If you enter a statement containing no LET-defined words, LET ECHO displays the statement as you entered it.
- If you enter a statement containing LET-defined words, LET ECHO displays the statement with the substitutions made.
- If the statement contains variable substitutions, LET ECHO displays the substitutions with the placeholders filled in.
- If the statement contains multiple-line substitutions, LET ECHO displays the statement with the substitutions on multiple lines.
- If the statement contains null substitutions, LET ECHO displays the statement with the LET-defined words deleted.
- If the statement contains recursive substitutions, the substitutions appear as they are finally resolved.
- LET ECHO may be coded at the top of a FOCEXEC. END ECHO gets coded on the last line of the FOCEXEC.

To turn off the LET ECHO facility and return to the FOCUS command level, enter:

```
ENDECHO
```

**Note:** If you enter a statement containing a variable substitution, you must enter as many words after the LET-defined word as there are placeholders in the phrase; otherwise, LET ECHO will wait for additional input.
Clearing LET Substitutions: LET CLEAR

Use the LET CLEAR command to clear LET substitutions.

**Syntax**

**LET CLEAR Command**

The syntax is

```
LET CLEAR {*
| word1  [word2...wordn} ...
```

where:

```
word1...wordn
```

Are the LET-defined words that you want cleared.

```
*
```

Clears all substitutions and global variables.

For example, to clear the words CHART, TESTNAME, and RIGHTNAME, enter:

```
LET CLEAR CHART TESTNAME RIGHTNAME
```

If there are no additional LETs in effect, the command

```
LET CLEAR *
```

would have the same effect.
Defining Word Substitutions: LET

Saving LET Equivalences in a File: LET SAVE

To save LET substitutions currently in effect, use the LET SAVE command.

Syntax

LET SAVE Command

The syntax is

LET SAVE [filename]

where filename is the eight-character name of the file in which you want to store the substitutions. If you do not supply a file name, the default file name is LETSAVE.

Since LET substitutions only last the duration of a FOCUS session, this command is helpful if you decide that you need the same substitutions for another session.

Defining PF Keys

In addition to being able to change parameters that govern your FOCUS environment, you can change the functions to which PF keys are set within specific FOCUS environments. You can define (or redefine) PF keys in the following environments: MODIFY, FIDEL, and TED.
6  FOCUS Query Commands

<table>
<thead>
<tr>
<th>Topics:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Listing Temporary Fields: ? DEFINE</td>
</tr>
<tr>
<td>• Displaying the File Directory Table: ? FDT</td>
</tr>
<tr>
<td>• Listing Data Source Statistics: ? FILE</td>
</tr>
<tr>
<td>• Listing HOLD Fields: ? HOLD</td>
</tr>
<tr>
<td>• Listing JOIN Structures: ? JOIN</td>
</tr>
<tr>
<td>• Displaying National Language Support: ? LANG</td>
</tr>
<tr>
<td>• Listing LET Substitutions: ? LET</td>
</tr>
<tr>
<td>• Displaying Explanations of Error Messages: ? n</td>
</tr>
<tr>
<td>• Querying Which PTFs Have Been Applied for a Specific Release</td>
</tr>
<tr>
<td>• Listing Parameter Settings: ? SET</td>
</tr>
<tr>
<td>• Listing Graph Parameters: ? SET GRAPH</td>
</tr>
<tr>
<td>• Listing Command Statistics: ? STAT</td>
</tr>
<tr>
<td>• Listing Global Variable Values: ? &amp;&amp;</td>
</tr>
</tbody>
</table>

FOCUS query commands provide information about your FOCUS environment. They display the status of FOCUS data sources, parameter settings, and temporary constructs: DEFINE fields, LET commands, HOLD files, USE directories, global variables, JOINs, and COMBINEs structures.

FOCUS returns the results of query commands as a comment in the HTML file. To view the query command results, use the Web browser functions to view the source of the HTML output.
FOCUS Query Commands

**Syntax**

**How to List Query Commands**

To list the query commands, just type a question mark (?) in a stored procedure. The syntax of the query command is:

```
? query [filename]
```

where:

- `query` is the subject of the query.
- `filename` is the name of the file that is the subject of the query. This parameter applies to only some queries.

**Reference**

**Query Command Summary**

Here is a list of query commands; a detailed description of each follows:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>? COMBINE</td>
<td>Lists FOCUS files comprising the current COMBINE structures.</td>
</tr>
<tr>
<td>? DEFINE</td>
<td>Lists currently active temporary fields created by DEFINE commands.</td>
</tr>
<tr>
<td>? F</td>
<td>Lists fields currently available to you.</td>
</tr>
<tr>
<td>? FDT</td>
<td>Shows physical attributes of FOCUS data sources.</td>
</tr>
<tr>
<td>? FF</td>
<td>Lists field names, aliases, and format information for an active Master File.</td>
</tr>
<tr>
<td>? FILE</td>
<td>Shows the number of segment instances in FOCUS data sources and the last time the data sources were changed.</td>
</tr>
<tr>
<td>? HOLD</td>
<td>Lists fields described in HOLD Master Files.</td>
</tr>
<tr>
<td>? JOIN</td>
<td>Lists JOIN structures between data sources.</td>
</tr>
<tr>
<td>? LET</td>
<td>Lists substitutions created with the LET command.</td>
</tr>
<tr>
<td>? LOAD</td>
<td>Provides information about all loaded files: the file type, file name and resident size.</td>
</tr>
<tr>
<td>? n</td>
<td>Displays explanations of error messages (n represents the number of the error message).</td>
</tr>
</tbody>
</table>
### Command Description

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>? PF[KEY]</td>
<td>Displays function key settings in the FIDEL environment.</td>
</tr>
<tr>
<td>? REL[LEASE]</td>
<td>Displays the FOCUS release number.</td>
</tr>
<tr>
<td>? SET GRAPH</td>
<td>Lists parameter settings controlling graphs displayed by FOCUS GRAPH requests.</td>
</tr>
<tr>
<td>? STAT</td>
<td>Displays statistics of the last FOCUS command executed.</td>
</tr>
<tr>
<td>? STYLE</td>
<td>Displays the current settings for page parameters.</td>
</tr>
<tr>
<td>? SU [userid</td>
<td>ddname]</td>
</tr>
<tr>
<td>? USE</td>
<td>Lists the FOCUS data sources specified in a USE command.</td>
</tr>
<tr>
<td>? &amp; &amp;</td>
<td>Displays values of global variables.</td>
</tr>
</tbody>
</table>

---

**Listing Temporary Fields: ? DEFINE**

The ? DEFINE command lists the currently active temporary fields created by either DEFINE commands or the DEFINE attribute in Master Files. ? DEFINE shows field names of up to 32 characters. If a name exceeds 32 characters, then a caret (>) in the 32nd position indicates a longer field name. The FIELDNO and ADDR columns are not displayed. The syntax is

```
? DEFINE [filename]
```

where:

- `filename` is the data source containing the values for the temporary fields. If `filename` is omitted, the command displays all temporary fields.
FOCUS Query Commands

The following information is listed for each DEFINEd field:

- **FILE**
  - Is the name of the data source containing the value for the temporary field.

- **FIELD NAME**
  - Is the name of the temporary field.

- **FORMAT**
  - Is the format of the temporary field. The notation is the same as that used for the FORMAT attribute in Master Files.

- **FIELDNO**
  - Is the number of the temporary field. The fields are numbered for each data source. The first number is two more than the number of data fields in the data source. For example, if there are 20 fields in a data source, the temporary fields for that data source are numbered from 22.

- **SEGMENT**
  - Is the number of the segment in the Master File containing `fieldname`. During reporting, FOCUS treats the temporary field as a field in this segment. To relate segment numbers to segment names, use `? FDT`.

- **VIEW**
  - Is the root segment of DEFINE that specifies an alternate view. For example:

    ```
    DEFINE FILE EMPLOYEE JOBCODE
    ```

- **TYPE**
  - Indicates whether the temporary field is defined by the Master File (MASTER) or by a DEFINE command (blank).

For example, issuing

```?
DEFINE
```

produces the following information:

```
FILE FIELD NAME FORMAT SEGMENT VIEW TYPE
EMPLOYEE PROJECTEDSAL   D12.2
EMPLOYEE FULLNAME       A26
>                        
```
Displaying the File Directory Table: ? FDT

The ? FDT command displays the file directory table, which lists the physical characteristics of a FOCUS data source. A FOCUS file is composed of fixed-length 4096-byte records called pages. Each segment and each index (those fields designated by the keyword FIELDTYPE=I in the Master File) occupies an integral number of pages. The file directory table shows the amount of space occupied by each segment instance in a page, the starting and ending page numbers, and the number of pages in between for each segment and index. A sample file directory table is also provided.

Syntax for ? FDT in a stored procedure is

```
? FDT filename
```

where:

filename

Is the name of the data source.

For example, issuing the command

```
? FDT EMPLOYEE
```

produced the following information:

```
DIRECTORY:EMPLOYEEFOCUS   F ON 09/25/1997 AT 09.50.28
DATE/TIME OF LAST CHANGE:   03/30/1999   16.19.22
SEGNAME   LENGTH   PARENT   START     END    PAGES  LINKS  TYPE
1  EMPINFO       22               1       1       1       6
2  FUNDTRAN      10       1       2       2       1       2
3  PAYINFO        8       1       3       3       1       3
4  JOBSEG        11       3                               4
5  SECSEG         4       4                               2
6  SKILLSEG      11       4                               2
7  ADDRESS       19       1       4       4       1       2
8  SALINFO        6       1       5       5       1       3
9  DEDUCT         5       8       6       8       3       2
10  ATTNDSEG       7       1                               3
11  COURSEG       11      10                               2
```
Following are explanations of each table column:

**SEGNAME**
Is the name of each segment in the file. The segments are numbered consecutively down the left. Unnumbered entries at the foot of the table are indexes. These indexes belong to fields having the attribute FIELDTYPE=I in the Master File.

**LENGTH**
Length of each segment instance in words (units of four bytes). To determine the number of instances that can fit into a page, divide this number into 992.

**PARENT**
Parent segment. The numbers refer to the segment numbers in the SEGNAME column.

**START**
Page number where the segment or index begins.

**END**
Page number where the segment or index ends.

**PAGES**
Number of pages occupied by the segment or index.

**LINKS**
Length of the pointer portion in each segment instance (in words). Every segment instance consists of two parts: data and pointers. Pointers are internal numbers that FOCUS uses to find other instances.

**TYPE**
Type of index. NEW indicates a binary index. OLD indicates a hash index.

**Note:** Segments of type KU, KM, DKU, DKM, KL, and KLU are not physically in this file; therefore, this information is omitted from the table.
Listing Data Source Statistics: ? FILE

The ? FILE command displays a chart showing the number of segment instances in a FOCUS data source and when the data source was last changed. The syntax in a stored procedure is

? FILE filename

where:

filename

Is the name of the data source.

For example, issuing the command

? FILE EMPLOYEE

produced the following:

<table>
<thead>
<tr>
<th>SEGNAME</th>
<th>ACTIVE COUNT</th>
<th>DELETED COUNT</th>
<th>DATE OF LAST CHG</th>
<th>TIME OF LAST CHG</th>
<th>LAST TRANS NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPINFO</td>
<td>12</td>
<td>12/21/93</td>
<td>11:01:32</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>FUNDTRAN</td>
<td>6</td>
<td>11/16/89</td>
<td>16:19:19</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>PAYINFO</td>
<td>19</td>
<td>11/16/89</td>
<td>16:19:20</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>21</td>
<td>11/16/89</td>
<td>16:19:21</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>SALINFO</td>
<td>70</td>
<td>11/16/89</td>
<td>16:19:22</td>
<td></td>
<td>448</td>
</tr>
<tr>
<td>DEDUCT</td>
<td>448</td>
<td>11/16/89</td>
<td>16:19:22</td>
<td></td>
<td>448</td>
</tr>
<tr>
<td>TOTAL SEGS</td>
<td>576</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL CHARS</td>
<td>8984</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL PAGES</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LAST CHANGE</td>
<td>01/29/96</td>
<td>11:01:32</td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

An explanation of each table column follows:

SEGNAME

Is the name of each segment in the data source. After the segments, the chart lists indexes (those fields specified by the keyword FIELDTYPE=I in the Master File).

ACTIVE COUNT

Number of instances of each segment.

DELETED COUNT

Number of segment instances deleted and space not re-used.

DATE OF LAST CHG

Date that data in a segment occurrence or index was last changed.

TIME OF LAST CHG

Time of day (on a 24-hour clock) last MODIFY of the file was executed for that segment/index.
FOCUS Query Commands

LAST TRANS NUMBER

Number of transactions performed by last MODIFY request or FSCAN session to access this segment. If the data source was changed under Simultaneous Usage mode or HLI, this column refers to the REF NUMB column of the CR HLIPRINT file.

If the segment was last modified by a MODIFY request reading data from a transaction file, this number indicates the last logical transaction record processed by the request. This is useful when you are restarting MODIFY requests that aborted in the middle of processing (see the MODIFY chapter in the Maintaining Databases manual).

The foot of the chart displays the following information:

TOTAL SEGS

Total number of segment instances in the file, and number of segments deleted when the file was last changed.

TOTAL CHARS

Number of characters of data in the file.

TOTAL PAGES

Number of pages in the data source. Pages are physical records in FOCUS data sources. Each page is 4096 bytes long.

LAST CHANGE

Date and time data source was last changed.

If a file is disorganized by more than 29% (that is, the physical placement of data in the data source is considerably different from its logical or apparent placement), the following message appears:

FILE APPEARS TO NEED THE -REBUILD- UTILITY
REORG PERCENT IS A MEASURE OF FILE DISORGANIZATION
0 PCT IS PERFECT -- 100 PCT IS BAD
REORG PERCENT IS x%

where:

x

Is a two- or three-digit percentage between 30 and 100.

The variable &FOCDISORG, used immediately after a ? FILE command, also indicates the level of disorganization. Following is a sample of how you can use &FOCDISORG in a -TYPE statement:

-IF &FOCDISORG GT 30 GOTO REBUILD
-TYPE THE AMOUNT OF DISORGANIZATION OF THIS FILE IS: &FOCDISORG

This statement, depending on the amount of disorganization, might produce the following message:

THE AMOUNT OF DISORGANIZATION OF THIS FILE IS: 10
If this message appears, you can improve file retrieval efficiency by rebuilding the file using the REBUILD command.

Note that when using a -TYPE statement with &FOCDISORG, a message is displayed even if the percentage of disorganization is less than 30%.

**Listing HOLD Fields: ? HOLD**

The ? HOLD command lists fields described in the Master File of a HOLD. The syntax in a stored procedure is

```
? HOLD [filename]
```

where:

*filename*

  Is the name assigned in the AS phrase when you created the HOLD file. If you omit *filename*, the file name defaults to HOLD.

The list displays the fields, their aliases, and their formats as defined by the USAGE or FORMAT attribute. The ? HOLD command can only display fields of HOLD files created during the current FOCUS request. ? HOLD displays field names of up to 32 characters. If a field name exceeds 32 characters, a caret (>) in the 32nd position indicates a longer field name.

For example, issuing the command from a HOLD file created with the COUNTRY and CAR fields of the CAR data source

```
? HOLD
```

produces the following information:

```
DEFINITION OF CURRENT HOLD FILE
FIELDNAME         ALIAS FORMAT
COUNTRY           E01     A10
CAR               E02     A16
> 
```
FOCUS Query Commands

Listing JOIN Structures: ? JOIN

The ? JOIN command lists the JOIN structures currently in effect. ? JOIN displays field names of up to 12 characters. If a field name exceeds 12 characters, then a caret in the 12th position indicates a longer field name.

For example, in a stored procedure

? JOIN

may produce the following information:

<table>
<thead>
<tr>
<th>JOINS CURRENTLY ACTIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST FIELD</td>
</tr>
<tr>
<td>FIELD</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>JOBCODE</td>
</tr>
</tbody>
</table>

Following are explanations of each table column:

- **HOST FIELD**: Is the name of the host field that is used to join the files.
- **FILE**: Is the name of the host file.
- **TAG**: Is a tag name that is used as a unique qualifier for field names in the host file.
- **CROSSREFERENCED FIELD**: Is the name of the cross-referenced field that is used to join the files.
- **FILE**: Is the name of the cross-referenced file.
- **TAG**: Is a tag name that is used as a unique qualifier for field names in the cross-referenced file.
- **AS**: Is the name of the joined structure.
- **ALL**: Shows Y for a non-unique join and a blank for a unique join.
Displaying National Language Support: ? LANG

The ? LANG command displays information on National Language Support, described in Chapter 4, Setting Parameters: SET. It displays the following information: language, code page, dollar value, and DBCS flags. For example, in a stored procedure

? LANG

produces a display similar to the following:

LANGUAGE AND DBCS STATUS

Language                01/AMENGLISH   (   )
Code Page               00037
Dollar value            5B($)       
DBCS Flag               OFF(SBCS)

Listing LET Substitutions: ? LET

The ? LET command lists the substitutions created by the LET command (described in Chapter 5, Defining Word Substitutions: LET), currently in effect. A sample list is also provided. The phrases in the left column are translated by FOCUS into the corresponding phrases in the right column. Note that phrase substitutions only last for the duration of the FOCUS session.

For example, in a stored procedure

? LET

may produce the following list:

PR       PRINT
TF       TABLE FILE EMPLOYEE
>
FOCUS Query Commands

Displaying Explanations of Error Messages: \( ? \ n \)

The \( ? \ n \) command displays a detailed explanation of an error message. This is useful when the online error message is not detailed enough to provide assistance in correcting the error. The syntax in a stored procedure is

\( ? \ n \)

where:

\( n \)

is the error message number.

When an error message does not exist or have an explanation, the query command only displays the text that the error message is missing. For example, if you receive the message

(FOC125) RECAP CALCULATIONS MISSING

and want to see an explanation, enter:

\( ? 125 \)

The following message is displayed:

(FOC125) RECAP CALCULATIONS MISSING
The word RECAP is not followed by a calculation. Either the RECAP should be removed, or a calculation provided.

Note: Error messages produced by certain FOCUS Interfaces, such as the DB2 and MODEL 204 Interfaces, are accessible through this utility.
Querying Which PTFs Have Been Applied for a Specific Release

By issuing `? PTF` at the FOCUS prompt, you are able to view a list of PTFs that have been applied to the version of FOCUS you are currently using. The following displays a screen with the `? PTF` command, followed by a sample result to the query.

```plaintext
> ? ptf
PTFS APPLIED TO RELEASE 70XFUC
FROM PTFTABLE LOCATED IN IBITEST LOADLIB C1

COUNT PTF NUM CREATED APPLIED SUPERSEDED BY PUT LEVEL
----- ------ ------ ------- ----------- -----------
1) 95828  ......... ......... 112500       ......       
2) 107164 ......... ......... 112500       ......       
3) 110763 ......... ......... 112500       ......       
4) 112600 19990427 19990513 ...... 200295

> 
```

Note: Dots are used to denote the lack of data if no information exists for a column entry in the resulting report. If there are no PTFs for the version of FOCUS that you are currently running, the following is displayed

```plaintext
NO PTFS HAVE BEEN APPLIED
```
FOCUS Query Commands

Listing Parameter Settings: ? SET

The ? SET command lists the parameter settings that control your FOCUS environment. FOCUS sets the default values for these parameters when you enter your FOCUS session, but you can change them with the SET command.

For example, in a stored procedure

\`\`? SET [ALL]\`

produces the following information:

<table>
<thead>
<tr>
<th>PARAMETER SETTINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL. OFF HIPERFOCUS OFF QUALCHAR .</td>
</tr>
<tr>
<td>ASNAMES FOCUS HOLDATTRS FOCUS QUALTITLES OFF</td>
</tr>
<tr>
<td>AUTOINDEX ON HOLDLIST ALL RECAP-COUNT OFF</td>
</tr>
<tr>
<td>AUTOPATH ON HOLDSTAT OFF SAVEMATRIX ON</td>
</tr>
<tr>
<td>BINS 64 HOTMENU OFF SCREEN ON</td>
</tr>
<tr>
<td>BLKCALC NEW INDEX TYPE NEW SHADOW PAGE OFF</td>
</tr>
<tr>
<td>BYPANELING OFF LANGUAGE AMENGLISH SPACES AUTO</td>
</tr>
<tr>
<td>CACHE 0 LINES/PAGE 66 SQLENGINE</td>
</tr>
<tr>
<td>CARTESIAN OFF LINES/PRINT 57 TCPIPINT OFF</td>
</tr>
<tr>
<td>CDN OFF MESSAGE ON TEMP DISK A</td>
</tr>
<tr>
<td>COLUMNSCROLL OFF MODE CMS TERMINAL IBM3270</td>
</tr>
<tr>
<td>DATETIME STARTUP/RESET NODATA . TITLES ON</td>
</tr>
<tr>
<td>DEFCENT 19 PAGE-NUM ON WIDTH 130</td>
</tr>
<tr>
<td>EMPTYREPORT OFF PANEL 0 WINPFKEY OLD</td>
</tr>
<tr>
<td>EXTSORT ON PAUSE ON XRETRIEVAL ON</td>
</tr>
<tr>
<td>FIELNAME NEW PRINT ONLINE YRTHRESH 0</td>
</tr>
<tr>
<td>FOCSTACK SIZE 8 PRINTPLUS ON</td>
</tr>
</tbody>
</table>

The ALL option displays all possible FOCUS SET parameters. All of the parameters are explained in Chapter 4, Setting Parameters: SET, which describes the SET command. Note that some of the parameters in this display are listed differently from the way you specify them in the SET command. These are:

- **LINES/PRINT**
  - Same as the LINES parameter.

- **LINES/PAGE**
  - Same as the PAPER parameter.

- **SESSION MONITOR**
  - Same as the SM parameter.

- **FOCSTACK SIZE**
  - Same as the FOCSTACK parameter.
INDEX TYPE
    Same as the INDEX parameter.

SHADOW PAGES
    Same as the SHADOW parameter.

In addition, the SET command lists the MODE parameter, which cannot be changed. This parameter indicates the operating system you are using. Values are:

CMS
TSO
CRJE (for MVS Batch)
MSO

Note: There is a SET parameter available with Dialogue Manager in FOCUS Release 7.0 that enables you to capture previously defined SET parameter values in amper variables.

Listing Graph Parameters: ? SET GRAPH

The ? SET GRAPH command lists the parameter settings that control graphs you produce with the FOCUS GRAPH command. These settings are described in Chapter 4, Setting Parameters: SET.

For example, issuing

? SET GRAPH

produces this information:

GRAPH PARAMETER SETTINGS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOTICK</td>
<td>ON</td>
<td>HISTOGRAM</td>
<td>ON</td>
</tr>
<tr>
<td>BARNUMB</td>
<td>OFF</td>
<td>HMAX</td>
<td>.00</td>
</tr>
<tr>
<td>BARSPACE</td>
<td>0</td>
<td>HMIN</td>
<td>.00</td>
</tr>
<tr>
<td>BARWIDTH</td>
<td>1</td>
<td>HSTACK</td>
<td>OFF</td>
</tr>
<tr>
<td>BSTACK</td>
<td>OFF</td>
<td>HTICK</td>
<td>.00</td>
</tr>
<tr>
<td>DEVICE</td>
<td>IBM3270</td>
<td>PIE</td>
<td>OFF</td>
</tr>
<tr>
<td>GMISSING</td>
<td>OFF</td>
<td>VAUTO</td>
<td>ON</td>
</tr>
<tr>
<td>GMISSVAL</td>
<td>.00</td>
<td>VAXIS</td>
<td>66</td>
</tr>
<tr>
<td>GPROMPT</td>
<td>OFF</td>
<td>VCLASS</td>
<td>.00</td>
</tr>
<tr>
<td>GRIBBON(GCOLOR)</td>
<td>OFF</td>
<td>VGRID</td>
<td>OFF</td>
</tr>
<tr>
<td>GRID</td>
<td>OFF</td>
<td>VMAX</td>
<td>.00</td>
</tr>
<tr>
<td>GTREND</td>
<td>OFF</td>
<td>VMIN</td>
<td>.00</td>
</tr>
<tr>
<td>HAUTO</td>
<td>ON</td>
<td>VTICK</td>
<td>.00</td>
</tr>
<tr>
<td>HAXIS</td>
<td>130</td>
<td>VZERO</td>
<td>OFF</td>
</tr>
<tr>
<td>HCLASS</td>
<td>.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

>
FOCUS Query Commands

If you change the PLOT parameter settings, a small table appears at the end of the list:

```
PLOT TABLE (EBCDIC):

ENTER PLOT MODE   0050 (FOR 3284 WIDTH)
EXIT PLOT MODE    0018 (FOR 3284 HEIGHT)
LEFT              0000
RIGHT             0000
UP                 0000
DOWN               0000
```

The entries in the table at the bottom are:

```
ENTER PLOT MODE
    Width of graph on IBM 3284 or 3287 printer.
EXIT PLOT MODE
    Height of graph on IBM 3284 or 3287 printer.
```

The parameters LEFT, RIGHT, UP, and DOWN should be ignored.

### Listing Command Statistics: ? STAT

The ? STAT command displays statistics for the last FOCUS command executed. For example, depending on the previous FOCUS commands executed, in a stored procedure

```
? STAT
```

produces the following information:

```
STATISTICS OF LAST COMMAND

RECORDS        =           0       SEGS DELTD     =           0
LINES          =           0       NOMATCH        =           0
BASEIO         =           0       DUPLICATES     =           0
SORTIO         =           0       FORMAT ERRORS  =           0
SORT PAGES     =           0       INVALID CONDTS =           0
READS          =           0       OTHER REJECTS  =           0
TRANSACTIONS   =           0       CACHE READS    =           0
ACCEPTED       =           0       MERGES         =           0
SEGS INPUT     =           0       SORT STRINGS   =           0
SEGS CHNGD     =           0       INDEXIO        =           0

INTERNAL MATRIX CREATED: YES        AUTOINDEX USED:               NO
SORT USED:              FOCUS        AUTOPATH USED:                NO
AGGREGATION BY EXT.SORT: NO          HOLD FROM EXTERNAL SORT:      NO
```
The statistics, explanations, and the applicable commands follow:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECORDS</td>
<td>Is for TABLE, TABLEF, MATCH, and GRAPH commands. Indicates the number of records used in the report. Note that the meaning of a record depends on the type of data source used.</td>
</tr>
<tr>
<td>LINES</td>
<td>Is for TABLE and TABLEF commands. Indicates the number of lines printed in a report.</td>
</tr>
<tr>
<td>BASEIO</td>
<td>Is for TABLE, TABLEF, GRAPH, MODIFY, and FSCAN commands. Indicates the number of I/O operations performed on the data source.</td>
</tr>
<tr>
<td>SORTIO</td>
<td>Is for TABLE, TABLEF, MATCH, and GRAPH commands. Indicates the number of I/O operations performed on the FOCSORT file, a work file invisible to the user.</td>
</tr>
<tr>
<td>SORTPAGES</td>
<td>Is for TABLE and TABLEF commands. Indicates the number of physical records in the FOCSORT file.</td>
</tr>
<tr>
<td>READS</td>
<td>Is for the MODIFY and FSCAN commands. Indicates the number of fixed format records read in external files by the FIXFORM command.</td>
</tr>
<tr>
<td>TRANSACTIONS</td>
<td>Is for the MODIFY and FSCAN commands. Indicates the number of transactions processed—inputs, updates, deletions, and rejections.</td>
</tr>
<tr>
<td>ACCEPTED</td>
<td>Is for the MODIFY and FSCAN commands. Indicates the number of transactions accepted.</td>
</tr>
<tr>
<td>SEGS INPUT</td>
<td>Is for MODIFY and FSCAN commands. Indicates the number of segment instances accepted into the data source.</td>
</tr>
<tr>
<td>SEGS CHNGD</td>
<td>Is for MODIFY and FSCAN commands. Indicates the number of segment instances updated in the data source.</td>
</tr>
<tr>
<td>SEGS DELTD</td>
<td>Is for MODIFY and FSCAN commands. Indicates the number of segment instances deleted from the data source.</td>
</tr>
<tr>
<td>NOMATCH</td>
<td>Is for the MODIFY command. Indicates the number of transactions rejected for lack of matching values in the data source. This occurs on an ON NOMATCH REJECT condition.</td>
</tr>
<tr>
<td>DUPLICATES</td>
<td>Is for the MODIFY command. Indicates the number of transactions rejected because their matching field values already exist in the data source. This occurs on an ON MATCH REJECT condition.</td>
</tr>
<tr>
<td>FORMAT ERRORS</td>
<td>Is for the MODIFY command. Indicates the number of transactions rejected because data field values for data fields do not conform to the field formats defined in the Master File.</td>
</tr>
<tr>
<td>INVALID CONDTS</td>
<td>Is for the MODIFY command. Indicates the number of transactions rejected because their values failed validation tests.</td>
</tr>
</tbody>
</table>
FOCUS Query Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER REJECTS</td>
<td>Indicates the number of transactions rejected for reasons other than listed above.</td>
</tr>
<tr>
<td>CACHE READS</td>
<td>Shows the number of CACHE READS performed (see Chapter 4, Setting Parameters: SET).</td>
</tr>
<tr>
<td>MERGES</td>
<td>Is the number of times that FOCUS merge routines have been invoked.</td>
</tr>
<tr>
<td>SORT STRINGS</td>
<td>Is the number of times that the FOCUS SORT capacity has been exceeded.</td>
</tr>
<tr>
<td>INTERNAL MATRIX CREATED</td>
<td>Can have a value of YES/NO.</td>
</tr>
<tr>
<td>SORT USED</td>
<td>Is the type of sort facility used. It can have a value of FOCUS, EXTERNAL, SQL, or NONE.</td>
</tr>
</tbody>
</table>

Listing Global Variable Values: ? &&

The ? && command lists the Dialogue Manager global variables and their current values. Global variables maintain their values for the duration of the FOCUS procedure (see Chapter 2, Managing Applications With Dialogue Manager). The variable names are on the left, their values are on the right.

For example, depending on the variables in effect, issuing the command in a stored procedure

```
? &&
```

may produce the following list:

```
&&STORECODE   001
&&STORENAME   MACYS
```

Your installation may choose to replace the ampersand (&) indicating Dialogue Manager variables with another symbol. In that case, use that symbol in your query command. For example, if your installation chooses the percent sign (%) to indicate Dialogue Manager variables, you can list global variables by entering:

```
? %
```

You can query Dialogue Manager variables from within the Dialogue Manager stored procedure by using the -? & command.
Session Tools

Session tools enable you to capture everything, line-for-line, that you input or output on the terminal. If you receive an error, you can check exactly what you typed in.

Terminal Operator Environment: The Save Output Window

The FOCUS Terminal Operator Environment is an optional window-oriented environment. It is easy to use and provides facilities which increase your productivity. For more information about this environment, see the Overview and Operating Environments manual.

Within TOE, 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 TABLE 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 the ENTER key to display the TABLE report in Hot Screen. Press the ENTER key 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.
7 Enhancing Application Performance

Topics:
- Introduction
- Loading Files: LOAD
- Compiling MODIFY Requests: The COMPILE Command
- File Transfer: Mainframe to Microcomputers
- New Method for Accessing FOCUS Data Sources (MVS Only): The MINIO Command

This chapter covers FOCUS facilities which are available to you across command environment boundaries. These facilities are easy to use and, in many cases, step-by-step instructions are provided.

Introduction

The FOCUS facilities discussed in this chapter are classified as file utilities for FOCUS and external files. They are summarized in the following table:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>Loads FOCUS procedures and Master Files into memory (see Loading Files: LOAD on page 7-2).</td>
</tr>
<tr>
<td>COMPILE</td>
<td>Translates MODIFY requests into compiled code ready for execution (see Compiling MODIFY Requests: The COMPILE Command on page 7-6).</td>
</tr>
<tr>
<td>XFER</td>
<td>Transfers files between mainframe computers and microcomputers (see File Transfer: Mainframe to Microcomputers on page 7-7).</td>
</tr>
<tr>
<td>MINIO</td>
<td><strong>Note:</strong> This facility is for MVS only. Improves performance by reducing I/O operations when accessing FOCUS data sources (see New Method for Accessing FOCUS Data Sources (MVS Only): The MINIO Command on page 7-10).</td>
</tr>
</tbody>
</table>
Loading Files: LOAD

Use the LOAD command to load the following types of files into memory for use within a FOCUS session:

- Master Files (MASTER).
- Access Files.
- FOCUS procedures (FOCEXC).
- Compiled MODIFY requests (FOCCOMP).
- MODIFY requests (MODIFY).

Using memory-resident files decreases execution time because the files do not have to be read from disk. Use the UNLOAD command to remove the files from memory (see UNLOAD Command Syntax on page 7-5).

LOAD Command Syntax

The syntax of the LOAD command is

LOAD filetype filename1 ... [filename2 ...]

where:

filetype
  Specifies the type of file to be loaded (MASTER, access file, FOCEXC, FOCCOMP, or MODIFY).

filename1...
  Specifies one or more files to be loaded. Separate the file type and file name(s) with a space.

For example, the following command loads the four FOCEXCs CARTEST, FOCEXMAP1, FOCEXMAP2, and FOCEXMAP3 into memory:

>LOAD FOCEXEC CARTEST FOCEXMAP1 FOCEXMAP2 FOCEXMAP3

A subsequent reference to one of these files during the current FOCUS session will use the loaded, rather than the disk, version.
Loading Considerations

The following are considerations for loading various types of files into memory.

Master Files (MASTER), FOCUS Procedures (FOCEXEC) and Access Files

Loading Master Files, Access Files, and FOCEXECs into memory eliminates the I/Os required to read them each time they are referenced. Whenever FOCUS requires a Master File, Access File, or executes a FOCEXEC, it first looks for a memory-resident MASTER, access file, or FOCEXEC file; if FOCUS cannot find the file in memory, it then searches for a disk version in the normal way.

Note:

- If you load a Master File, Access File, or a FOCEXEC that has already been loaded into memory, the new copy replaces the old copy.
- Do not load a Master File, Access File, or a FOCEXEC that you are developing, because FOCUS will always use the memory-resident copy of the file (until you reload it), rather than the one you are developing. This is because the copy that you are developing on TED or your system editor is the disk copy, not the memory-resident copy.
- A loaded Master File, Access File, or FOCEXEC requires a maximum of 80 bytes of memory for each of its records plus a small amount of control information, rounded up to a multiple of 4200 bytes.
- The following are the file types for the various Access Files:

<table>
<thead>
<tr>
<th>Access File</th>
<th>File type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADABAS</td>
<td>FOCADBS</td>
</tr>
<tr>
<td>DATACOM</td>
<td>FOCDTCM</td>
</tr>
<tr>
<td>DB2</td>
<td>FOCSQL</td>
</tr>
<tr>
<td>IDMS</td>
<td>FOCIDMS</td>
</tr>
<tr>
<td>IMS (IMS=NEW only)</td>
<td>ACCESS</td>
</tr>
<tr>
<td>MODEL 204</td>
<td>FOCM204</td>
</tr>
<tr>
<td>ORACLE</td>
<td>FOCSQL</td>
</tr>
<tr>
<td>SQLDS</td>
<td>FOCSQL</td>
</tr>
<tr>
<td>S2K</td>
<td>FOCS2K</td>
</tr>
<tr>
<td>SUPRA</td>
<td>ACCESS</td>
</tr>
<tr>
<td>TERADATA</td>
<td>FOCSQL</td>
</tr>
<tr>
<td>TOTAL</td>
<td>FOCTOTAL</td>
</tr>
</tbody>
</table>
Enhancing Application Performance

Compiled MODIFY Requests (FOCCOMP)

When you load a compiled MODIFY request, FOCUS loads the FOCCOMP file from disk into memory, then reads and parses the Master File and binds the description to the FOCCOMP file. You may then run the request by issuing the RUN command

```
RUN request
```

where `request` is the name of the compiled request stored in memory. The RUN command causes FOCUS to search for a memory-resident FOCCOMP file. If FOCUS cannot find the file, it searches for a disk version in the normal way.

Loading FOCCOMP files not only eliminates the I/Os required to read large FOCCOMP files and their associated Master Files, but also causes another, more subtle effect. When you issue the RUN command to execute a FOCCOMP file from disk, virtual storage must be paged in to accommodate it. If the FOCCOMP file is large, it may require many pages (and a large virtual storage area) in a very short time. If you load the FOCCOMP file first, the initial surge of paging occurs only once at LOAD time. After that, each execution of the loaded file requires a lower paging rate.

MODIFY Requests (MODIFY)

The LOAD MODIFY command is similar to the COMPILE command (described in the Maintaining Databases manual) except that instead of writing the compiled output to a FOCCOMP file on disk, FOCUS writes the output into memory as a pre-loaded, compiled MODIFY. FOCUS then reads the Master File associated with the MODIFY statement from disk and translates it into an internal table that is tightly bound with the compiled MODIFY. Thus the statement

```
>LOAD MODIFY NEWTAX
```

has substantially the same effect as

```
>COMPILE NEWTAX
>LOAD FOCCOMP NEWTAX
```

except that the compiled code is never written to disk.

After you enter a LOAD MODIFY command, the resulting compiled MODIFY is indistinguishable from code loaded with LOAD FOCCOMP. Thus the UNLOAD MODIFY and ? LOAD MODIFY commands produce exactly the same results as the UNLOAD FOCCOMP and ? LOAD FOCCOMP commands. (The UNLOAD and ? LOAD commands are discussed in UNLOAD Command Syntax on page 7-5 and Listing Loaded Files on page 7-5.)

Note that the UNLOAD FOCCOMP and UNLOAD MODIFY commands unload the bound Master File as well.

When you issue the RUN command to invoke a MODIFY procedure, FOCUS looks for a memory-resident compiled procedure (created by a LOAD FOCCOMP or LOAD MODIFY command) of that name. If the procedure cannot be found, FOCUS then searches for a disk version of the FOCCOMP file in the normal way.
The benefits of the LOAD MODIFY command are that disk space is not used to store the FOCCOMP file, disk I/Os are reduced, the FOCEXC cannot get out of step with the compiled version, and the paging rate is reduced as it is with FOCCOMP files.

**UNLOAD Command Syntax**

Use the UNLOAD command to remove files from memory. The syntax of the UNLOAD command is

\[
\text{UNLOAD} \quad [\ast|\text{filetype}] \quad [\ast|\text{filename1}... \text{filename2}...]
\]

where:

- **filetype**
  - Specifies the type of file to be unloaded (MASTER, access file, FOCEXC, MODIFY, or FOCCOMP). To unload all files of all types, use an asterisk.

- **filename1...**
  - Specifies one or more files to be unloaded. Separate the file type and file name(s) with a space. To unload all files of that file type, use an asterisk.

For example, the following command unloads the two memory-resident FOCEXECs CARTEST and FOCMAP3:

\[
>\text{UNLOAD FOCEXEC CARTEST FOCMAP3}
\]

Any subsequent reference to one of these files will use the disk version.

**Listing Loaded Files**

The query command ? LOAD displays the file type, file name, and resident size of currently loaded files. Specify the query command as follows

\[
? \text{LOAD} \quad [\text{filetype}]
\]

where:

- **filetype**
  - Specifies the type of file (MASTER, FOCEXC, access file, FOCCOMP, or MODIFY) on which information will be displayed. To display information on all memory-resident files, omit file type.

For example, entering the following command

\[
? \text{LOAD}
\]

lists all files of all types that are currently loaded into memory:

<table>
<thead>
<tr>
<th>FILES CURRENTLY LOADED</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAR         MASTER 4200  BYTES</td>
</tr>
<tr>
<td>EXPERSON    MASTER 4200  BYTES</td>
</tr>
<tr>
<td>CARTEST     FOCEXEC 8400  BYTES</td>
</tr>
</tbody>
</table>
Compiling MODIFY Requests: The COMPILE Command

The COMPILE command translates a MODIFY request stored in a FOCEXEC into an executable code module. This module, like an object code module, cannot be edited by a user. However, it loads faster than the original request because the MODIFY statements have already been interpreted by FOCUS (the initialization time of a compiled MODIFY module can be four to ten times faster than the original request). Compiling a request can save a significant amount of time if the request is large and must be executed repeatedly. You compile the request once, and execute the module as many times as you need it.

Enter the COMPILE command at the FOCUS command level (the FOCUS prompt). The syntax is:

```
COMPILE focexec [AS module]
```

where:

- `focexec` is the name of the FOCEXEC where the request is stored.
- `module` is the name of the module. The default is the FOCEXEC name.

FOCEXEC names and module names are system dependent.

To execute a module, enter from the FOCUS command level:

```
RUN module
```

where `module` is the name of the module. You will see no difference in execution between the module and the original request, but it will load much faster.

Please note the following regarding compilation of MODIFY requests:

- The FOCEXEC procedure to be compiled may only contain one MODIFY request. It may not contain any other FOCUS, Dialogue Manager, or operating system statements.
- Before compiling a request or executing a module, allocate all input and output files such as transaction files and log files. These allocations must be in effect at run time.
- Before compilation, issue any SET, USE, COMBINE, or JOIN statements necessary to run the request.
- If the data source you are modifying is joined to another file (using the JOIN command) during compilation, it must be joined to the file at run time.
- If you are modifying a combined structure (using the COMBINE command), the structure must be combined both at compilation and at run time.
- FOCEXECs prompt for Dialogue Manager variable values at compilation time. These values cannot be changed at run time.
- If you are using FOCUS security to prevent unauthorized users from executing the request, the password you set at compilation time must be the same one set at run time.
FOCUS has several facilities to enable you to transfer EBCDIC and ASCII files between mainframes, minicomputers, and microcomputers. This ability can be very useful for a variety of reasons:

- You might develop a database system on a microcomputer, but want to upload it to a mainframe computer.
- Your company may be changing from one computer environment to another.
- You may want to download data from a mainframe to a personal computer, where you can use this data with LOTUS 1-2-3, MULTIPLAN, or a word-processing package. After you modify the data at the microcomputer level, you can then upload the data, and update the original mainframe file.
- Use FOCUS as your general communication program for all your system communication needs.

There are two ways you can download or upload a file:

- You can use the XFER command, which can transfer any type of EBCDIC or ASCII file (such as data sources, Master Files, or extract files) between a mainframe and a minicomputer or microcomputer.
- You can also use the PCHOLD option (which downloads extracted data to a microcomputer).

**Communication Requirements**

Before you can transfer data between a mainframe and a microcomputer, you must have:

- FOCUS Report Writer and the FOCUS PC Data Export Interface on the mainframe.
- PC/FOCUS on the microcomputer.
- A microcomputer system with a working asynchronous modem or a 3278 coaxial connection using an IRMA, PCOX, or FORTE board.
- An established link between the mainframe and microcomputer. Your microcomputer must emulate a terminal. (For information on linking to the mainframe from a microcomputer, see the PC/FOCUS Utilities Manual.)

It is also important to understand that only EBCDIC and ASCII sequential files can be transferred between microcomputers and mainframes. If you have a non-sequential file (a FOCUS file, for instance), this file must be converted to a sequential file. FOCUS files can be converted using the ON TABLE SAVE command. For example:

```
TABLE FILE SALES
PRINT SEG.STORE_CODE AND SEG.DATE AND SEG.PROD_CODE
ON TABLE SAVE
END
```
The SAVE option automatically converts the extracted data into a sequential file which you can download to a microcomputer using the XFER command (for more information on SAVE files, see the Creating Reports manual). After transferring the file, you can load the data back into a FOCUS file using a PC/FOCUS MODIFY request.

You can also use the syntax

```
ON TABLE HOLD FORMAT ALPHA
```

to commit the data directly to the target format. For example:

```
TABLE FILE SALES
PRINT *
ON TABLE HOLD FORMAT ALPHA
END
```

**Downloading and Uploading Files: The XFER Command**

The XFER command enables you to download and upload files between a mainframe and a microcomputer. For information on linking procedures, see the PC/FOCUS Utilities Manual.

You issue the XFER command at the FOCUS command level (at the prompt) on the mainframe.

**Note:** Before you use the XFER command, you must establish a link between the microcomputer and the mainframe and issue a FILEDEF, ALLOCATE, or DYNAM ALLOCATE command.

**Downloading Files**

The syntax to download is

```
XFER ddbname TO d:fn.ext
```

where:

- **ddname**
  - Is the ddbname from a FILEDEF, ALLOCATE, or DYNAM ALLOCATE command.

- **d:fn.ext**
  - Stands for the disk destination, the name of the file you want on the PC, and its extension.
  - Make sure you specify the appropriate extension. For instance, if you want to transfer a Master File to the microcomputer, enter the following:

```
XFER ddbname TO C:EXAMPLE.MAS
```

For a list of file name extensions, see the PC/FOCUS Utilities Manual.
For example:

```
XFER SAVE TO C:NEW.DAT
```

After you press the ENTER key, the following message will appear:

```
V2.0 XFRING FROM TO PC FILE NEW.DAT
```

The data is available for use on the microcomputer, accessible by the file name NEW.DAT on the C drive.

Note: If you issue the command XFER, FOCUS will prompt you for the rest of the information.

### Uploading Files

The syntax to upload a file is

```
XFER ddbname FROM d:fn.ext
```

where:

- **ddname**
  
  Is the ddbname from a FILEDEF or ALLOCATE command. You must specify DCB parameters in the FILEDEF and the ALLOCATE commands first.

- **d:fn.ext**
  
  Stands for the disk location, the name of the file, and its extension.

For example, to transfer a Master File from the microcomputer, enter the following:

```
XFER EXAMPLE FROM C:EXAMPLE.MAS
```

The file is then accessible on the mainframe with the file name used in the FILEDEF or ALLOCATE command.

Note: If you issue the XFER command with no operands, FOCUS prompts you for the rest of the information.

Information Builders recommends using FTP to transfer files from the Mainframe to any other system. This requires TCP/IP on your MVS/CMS installation.
New Method for Accessing FOCUS Data Sources (MVS Only): The MINIO Command

MINIO is a new I/O buffering technique that improves performance by reducing I/O operations when accessing FOCUS data sources under MVS. With MINIO set on, no block is ever read more than once, and therefore the number of reads performed will be the same as the number of tracks present. This results in an overall reduction in elapsed times when reading and writing.

The syntax is

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

where:

- **ON**
  - Enables MINIO. ON is the default.

- **OFF**
  - Disables MINIO.

With FOCUS data sources that are not disorganized, MINIO can greatly reduce the number of I/O operations for TABLE and MODIFY commands. I/O reductions of up to fifty percent are achievable with MINIO. The actual reduction will vary depending on data source structure and average numbers of children segments per parent segment. By reducing I/O operations, elapsed times for TABLE and MODIFY commands also drop.

**MINIO Usage**

MINIO reduces CPU time slightly while slightly raising memory utilization. MINIO requires one track I/O buffer per referenced segment type. Between 40K and 48K of above-the-line virtual memory is needed per referenced segment.

When MINIO is enabled, FOCUS decides for each command whether or not to employ it, and which data sources to use it with. It is possible in executing a single command referencing several data sources that MINIO might be used for some but not for others. Data sources accessed via indexes, or physically disordered through online updates, are not candidates for MINIO buffering. Physical disorganization, in this case, means that the sequence of selected records jumps all over the data source, as opposed to progressing steadily forward. When disorganization occurs, MINIO abandons its buffering techniques and resorts to the standard I/O methodology.

When reading data sources, MINIO is used with TABLE, TABLEF, GRAPH, MATCH and during the DUMP phase of the REBUILD command, provided the target data source is not accessed via an index or is physically disorganized.
New Method for Accessing FOCUS Data Sources (MVS Only): The MINIO Command

When writing to data sources, MINIO is used with MODIFY but never with MAINTAIN, provided there is no CRTFORM or COMMIT subcommand. CRTFORMs indicate online transaction processing, which requires that completed transactions be written out to the data source. COMMITs are explicit orders to do so. These events are incompatible with MINIO minimization logic and therefore rule out its use.

As with reads, using MINIO with MODIFY also requires that a data source be accessed sequentially. Attempts to access an index, or update physically disorganized data sources both cause MINIO to be disabled. In addition, frequent repositioning to previously accessed records, even within well-organized data sources, will cause MINIO to be disabled. The ? STAT command is used to determine whether the previous data source access command employed MINIO. Typing ? STAT generates a screen similar to the following:

```
STATISTICS OF LAST COMMAND
RECORDS  = 0  SEGS CHNGD  = 0
LINES    = 0  SEGS DELTD  = 0
BASEIO   = 87  NOMATCH     = 0
TRACKIO  = 16  DUPLICATES = 0
SORTIO   = 0  FORMAT ERRORS= 0
SORT PAGES = 0  INVALID CONDTS = 0
READS    = 1  OTHER REJECTS = 0
TRANSACTIONS = 1500  CACHE READS = 0
ACCEPTED = 1500  MERGES      = 0
SEGS INPUT = 1500  SORT STRINGS = 0
INTERNAL MATRIX CREATED: YES
SORT USED:       FOCUS
MINIO USED:      YES
AUTOINDEX USED:  NO
AUTOPATH USED:   NO
```

In the preceding example MINIO USED is displayed as YES. It may also display NO or DISABLED.

- **YES** means that MINIO buffering has taken place reducing the number of tracks read/written to the FOCUS data source.
- **NO**, means that MINIO buffering has not taken place.
- **DISABLED** means that MINIO buffering was started but terminated as no performance gains could be made. This does not mean that the command did not complete successfully. It only indicates that MINIO buffering began and ended during the read/write.
Restrictions With MINIO

Note the following restrictions when you are using the MINIO command:

- When MINIO is used with MODIFY, all CHECK subcommands are ignored. If a MODIFY command terminates abnormally, the condition of the data source is unpredictable, and it should be restored from a backup copy and the update repeated. Since MINIO is designed to minimize I/O during large data source loads and updates, it has no checkpoint or restart facility. If this is unacceptable, set MINIO off.

- MINIO is not used to access data sources through FOCUS Database Servers (formerly called sink machines) or HLI programs.

- MINIO requires the presence of the TRACKIO feature. Meaning, TRACKIO must be set to ON which is the default setting. If TRACKIO is set to OFF, then MINIO is deactivated.

- MINIO buffering starts when the FOCUS data source exceeds 64 pages in size. If this size is never reached, MINIO is never activated.

- If the file being modified UPDATEs, INCLUDEs, or DELETEs a field that is indexed, MINIO is disabled. In other words, FIELDTYPE=1 or INDEX=1 is coded in the Master File for this field.

- CRTFORM and COMMIT commands disable MINIO.

- MAINTAIN procedures will not use MINIO buffering techniques.

- MINIO is not enabled if the data source is physically disorganized by transaction processing.
8 Euro Currency Support

Topics:
- Converting Currencies
- Preparing FOCUS to Process Currency Conversions
- Activating the Currency Database
- Processing Currency Data

With the introduction of the euro currency, businesses need to maintain books in two currencies, add new fields to their database designs, and perform new types of currency conversions. FOCUS can perform currency conversions according to the rules specified by the European Union. Before you can use FOCUS to process currency conversions, you must:

- Create a currency database with the currency IDs and exchange rates you will use. See Creating the Currency Database on page 8-3.
- Identify fields in your data sources that represent currency data. See Identifying Fields That Contain Currency Data on page 8-5.
- Activate your currency database. See Activating the Currency Database on page 8-7.

After you complete these preliminary steps, you can perform currency conversions. See Processing Currency Data on page 8-9.

Note: Operating system vendors are in the process of integrating the euro currency symbol into their environments. As the euro symbol becomes available, FOCUS will support it.

Converting Currencies

Although the euro was introduced in 11 countries of the European Union on January 1, 1999, it will not immediately replace local currencies in those countries. During the transition period from 1999 to 2002, both traditional currencies and the euro will be used simultaneously for accounting purposes and non-cash transactions in each participating country. Euro cash will not be introduced until January 1, 2002, and by July 1, 2002 the traditional currencies will no longer be legal tender.

On the last day of 1998, the European Union set fixed exchange rates between the euro and the traditional national currency in each of the 11 adopting member nations. While the exchange rates within “Euroland” will remain fixed, exchange rates between the euro and non-euro countries will continue to vary freely and, in fact, several rates may be in use at one time (for example, actual and budgeted rates).
The European Union has established the following rules for currency conversion:

- The exchange rate must be specified as a decimal value, $r$, with six significant digits (not six decimal places). For example, 123.456 has six significant digits but not six decimal places. This rate will establish the following relationship between the euro and the particular national currency:

  \[ 1 \text{ euro} = r \text{ national units} \]

- To convert from the euro to the national unit, multiply by $r$ and round the result to two decimal places.

- To convert from the national currency to the euro, divide by $r$ and round the result to two decimal places.

- To convert from one national currency to another, first convert from one national unit to the euro, rounding the result to at least three decimal places (FOCUS rounds to exactly three decimal places). Then convert from the euro to the second national unit, rounding the result to two decimal places. The following diagram illustrates this two-step conversion process known as *triangulation*:

**Converting 10 US Dollars to French Francs**

\[ r = 1.00000 \]

**Step 1:**

\[ 10 \text{ USD} = \frac{10}{1.17249} \text{ EUR} = 8.529 \text{ EUR} \]

**Step 2:**

\[ 8.344 \text{ EUR} = 8.529 \times 6.55957 \text{ FRF} = 55.95 \text{ FRF} \]

\[ r = 1.17249 \quad \quad r = 6.55957 \]
Preparing FOCUS to Process Currency Conversions

Although 11 or more currencies in the European Union will be converting to the euro, more than 100 currencies have a recognized status worldwide. In addition, you may need to define custom currencies for some applications.

You identify your currency codes and rates by creating a currency database. The currency database can be any type of data source that FOCUS can access.

Creating the Currency Database

For each type of currency you need, you must supply the following values in your currency database:

- A three-character code to identify the currency, such as USD for U.S. dollars or BEF for Belgian francs. (For a partial list of recognized currency codes, see Sample Currency Codes on page 8-12.)

- One or more exchange rates for the currency.

There is no limit to the number of currencies you can add to your currency database; the currencies you can define are not limited to official currencies and, therefore, the currency database can be fully customized for your applications.

We strongly recommend that you create a separate database for the currency data rather than adding the currency fields to another data source. A separate currency database enhances performance and minimizes resource utilization because FOCUS loads the currency database into memory before you perform currency conversions.
Euro Currency Support

**Syntax** How to Specify Currency Codes and Rates in a Master File

The currency database can be any type of data source accessible by FOCUS (for example, FOCUS, FIX, DB2, or VSAM). The currency Master File must have one field that identifies each currency ID you will use and one or more fields to specify the exchange rates.

The syntax is

```plaintext
FIELD  = CURRENCY_ID,   FORMAT = A3,                     ACTUAL = A3 ,$
FIELD  = rate1 ,         FORMAT = {D12.6|numeric_format1}, ACTUAL = A12,$
  .
  .
FIELD  = raten ,         FORMAT = {D12.6|numeric_formatn}, ACTUAL = A12,$
```

where:

- **CURRENCY_ID**
  - Is the required field name. The values stored in this field are the three-character codes that identify each currency, such as USD for U.S. dollars. Each currency ID can be a universally recognized code or a user-defined code.
  - **Note:** FOCUS automatically recognizes the code EUR; you should not store this code in your currency database. See Sample Currency Codes on page 8-12 for a list of common currency codes.

- **rate1, ..., raten**
  - Are types of rates (such as BUDGET, FASB, ACTUAL) to be used in currency conversions. Each rate is the number of national units that represent one euro.

- **numeric_format1, ..., numeric_formatn**
  - Are the display formats for the exchange rates. Each format must be numeric. The recommended format, D12.6, ensures that the rate is expressed with six significant digits as required by the European Union conversion rules. Do not use Integer format (I).

- **ACTUAL An**
  - Is required only for non-FOCUS data sources.

**Note:** The maximum number of fields in the currency database must not exceed 255 (that is, the CURRENCY_ID field plus 254 currency conversion fields).
Preparing FOCUS to Process Currency Conversions

**Example** Specifying Currency Codes and Rates in a Master File

The following Master File for a comma-delimited currency database specifies two rates for each currency, ACTUAL and BUDGET:

```
FILE = CURRCODE, SUFFIX = COM, $
FIELD = CURRENCY_ID, FORMAT = A3, ACTUAL = A3 ,$
FIELD = ACTUAL, ALIAS =, FORMAT = D12.6, ACTUAL = A12 ,$
FIELD = BUDGET, ALIAS =, FORMAT = D12.6, ACTUAL = A12 ,$
```

The following is sample data for the currency database defined by this Master File:

```
FRF,  6.55957,  6.5000,$
USD,  1.17249,  1.20000,$
BEF,  40.3399, 41.00000,$
```

Identifying Fields That Contain Currency Data

Once you have created your currency database, you must identify the fields in your data sources that represent currency values. To designate a field as a currency-denominated value (a value that represents a number of units in a specific type of currency) add the CURRENCY attribute to one of the following:

- The FIELD specification in the Master File.
- The left side of a DEFINE or COMPUTE.
Euro Currency Support

**Syntax** How to Identify a Currency Value

Use the following syntax to identify a currency-denominated value:

- **In a Master File**
  
  ```
  FIELD = currfield, FORMAT = numeric_format, , CURR = {curr_id|codefield} ,$
  ```

- **In a DEFINE in the Master File**
  
  ```
  DEFINE currfield/numeric_format CURR curr_id = expression ;$
  ```

- **In a DEFINE FILE command**
  
  ```
  DEFINE FILE filename
  currfield/numeric_format CURR curr_id = expression ;
  END
  ```

- **In a COMPUTE command**
  
  ```
  COMPUTE currfield/numeric_format CURR curr_id = expression ;
  ```

where:

- `filename` is the name of the file for which this field is defined.
- `currfield` is the name of the currency-denominated field.
- `numeric_format` is a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. Do not use I or F format.
- `CURR` indicates that the field value represents a currency-denominated value. CURR is an abbreviation of CURRENCY, which is the full attribute name.
- `curr_id` is the three-character currency ID associated with the field. In order to perform currency conversions, this ID must either be the value EUR or match a CURRENCY_ID value in your currency database.
codefield

Is the name of a field, qualified if necessary, that contains the currency ID associated with currfield. The code field should have format A3 or longer and is interpreted as containing the currency ID value in its first three bytes. For example:

\[
\text{FIELD = PRICE, FORMAT = P12.2C, \ldots, CURR = TABLE.FLD1,}\$
\]

\[
\text{\ldots}
\]

\[
\text{FIELD = FLD1, FORMAT = A3, \ldots,}\$
\]

The field named FLD1 contains the currency ID for the field named PRICE.

does not appear in the document.

evaluation  Identifying a Currency-Denominated Field

Assume that the currency database contains the currency ID value BEF (Belgian francs).

If the FINANCE data source contains a field named PRICE that is denominated in Belgian francs, the description of PRICE in the FINANCE Master File could be:

\[
\text{FIELD = PRICE, ALIAS=, FORMAT = P17.2, CURR=BEF,}\$
\]

Activating the Currency Database

Before you can perform currency conversions, you must bring the relevant currency database into memory by issuing the SET EUROFILE command.

Syntax

How to Activate Your Currency Database

Issue the following command at the FOCUS command prompt, in a FOCEXEC, or in any supported profile:

\[
\text{SET EUROFILE = \{ddname|OFF}\}}
\]

where:

\[
\text{ddname}
\]

Is the name of the Master File for the currency database. There is no default value for EUROFILE. The ddname must refer to a data source known to FOCUS and accessible by FOCUS in read-only mode.

\[
\text{OFF}
\]

Deactivates the currency database and removes it from memory.

During your FOCUS session, if you want to access a different currency database, you can re-issue the SET EUROFILE command.
Euro Currency Support

Note:

- You cannot append any additional SET parameters to the SET EUROFILE command line. For example, the PAUSE setting would be lost if you issued the following command:

  `SET EUROFILE=filename , PAUSE=OFF`

- You cannot issue the SET EUROFILE command within a TABLE request.

Syntax  How to Determine the Currency Database in Effect

If you want to determine which currency database is in effect, issue the `SET ALL` command or the new EUROFILE query command:

`? SET EUROFILE`

Example  Determining the Currency Database in Effect

Assume the currency database is named CURRCODE.

If you issue the following commands:

```
set eurofile = currcode
? set eurofile
```

FOCUS returns the following response:

```
EUROFILE            CURRCODE
```

Reference  SET EUROFILE Error Messages and Notes

Issuing the SET EUROFILE command when the currency database Master File does not exist generates the following error message:

`(FOC205) THE DESCRIPTION CANNOT BE FOUND FOR FILE NAMED: ddname`

Issuing the SET EUROFILE command when the currency Master File specifies a FOCUS database and the associated FOCUS database does not exist generates the following error message:

`(FOC036) NO DATA FOUND FOR THE FOCUS FILE NAMED: name`

Note for Pooled Table users: The SET EUROFILE command creates a pool boundary.
Processing Currency Data

After you have created your currency database, identified the currency-denominated fields in your data sources, and activated your currency database, you can perform currency conversions.

Each currency ID in your currency database generates a virtual conversion function whose name is the same as its currency ID. For example, if you added BEF to your currency database, a virtual BEF currency conversion function will be generated.

The euro function, EUR, is supplied automatically with FOCUS. You do not need to add the EUR currency ID to your currency database.

Syntax

How to Convert Currency Data

Use the following syntax for calling a currency conversion function:

- In a TABLE, GRAPH, or MODIFY procedure:

  ```
  DEFINE FILE filename
  result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);
  END
  
  or

  COMPUTE result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);
  ```

- In a Master File:

  ```
  DEFINE result/format [CURR curr_id] = curr_id(infield, rate1 [,rate2]);$
  ```

where:

- `filename` is the name of the file for which this field is defined.

- `result` is the converted currency value.

- `format` must be a numeric format. Depending on the currency denomination involved, the recommended number of decimal places is either two or zero. Do not use I or F format. The result will always be rounded to two decimal places, which will display if the format allows at least two decimal places.
Euro Currency Support

**curr_id**
Is the currency ID of the result field. This ID must be the value EUR or match a currency ID in your currency database; any other value generates the following message:

(FOC263) EXTERNAL FUNCTION OR LOAD MODULE NOT FOUND: curr_id

**Note:** The CURR attribute on the left side of the DEFINE or COMPUTE identifies the result field as a currency-denominated value which can be passed as an argument to a currency function in subsequent currency calculations. Adding this attribute to the left side of the DEFINE or COMPUTE does not invoke any format or value conversion on the calculated result.

**infield**
Is a currency-denominated value. This input value will be converted from its original currency to the `curr_id` denomination. If the `infield` and `result` currencies are the same, no calculation is performed and the `result` value is the same as the `infield` value.

**rate1**
Is the name of a rate field from the currency database. The `infield` value is divided by its currency’s `rate1` value to produce the equivalent number of euros.

If `rate2` is not specified in the currency calculation and triangulation is required, this intermediate result is then multiplied by the `result` currency’s `rate1` value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for `rate1` as long as it indicates the number of units of the `infield` currency denomination that equals one euro.

**rate2**
Is the name of a rate field from the currency database. This argument is only used for those cases of triangulation in which you need to specify different rate fields for the `infield` and `result` currencies. It is ignored if the euro is one of the currencies involved in the calculation.

The number of euros that was derived using `rate1` is multiplied by the `result` currency’s `rate2` value to complete the conversion.

In certain cases, you may need to provide different rates for special purposes. In these situations you can specify any field or numeric constant for `rate2` as long as it indicates the number of units of the `result` currency denomination that equals one euro.

**Note:** Maintain does not support these currency conversion functions.
Example  Converting Currencies

Assume that the currency database contains the currency IDs USD and BEF, and that PRICE is
denominated in Belgian francs as follows:

FIELD = PRICE, ALIAS=, FORMAT = P17.2, CURR=BEF,$

• The following example converts PRICE to euros and stores the result in PRICE2 using the
  BUDGET conversion rate for the BEF currency ID:

  COMPUTE PRICE2/P17.2 CURR EUR = EUR(PRICE, BUDGET);

• This example converts PRICE from Belgian francs to US dollars using the triangulation
  rule:

  DEFINE PRICE3/P17.2 CURR USD = USD(PRICE, ACTUAL);$

  First PRICE is divided by the ACTUAL rate for Belgian francs to derive the number of
  euros rounded to three decimal places. Then this intermediate value is multiplied by the
  ACTUAL rate for US dollars and rounded to two decimal places.

• The following example uses a numeric constant for the conversion rate:

  DEFINE PRICE4/P17.2 CURR EUR = EUR(PRICE,5);$

• The next example uses the ACTUAL rate for Belgian francs in the division and the
  BUDGET rate for US dollars in the multiplication:

  DEFINE PRICE5/P17.2 CURR USD = USD(PRICE, ACTUAL, BUDGET);$

Reference  Currency Calculation Processing and Messages

The result is always calculated with very high precision, 31 to 36 significant digits, depending
on platform. The precision of the final result is always rounded to two decimal places. In order
to display the result to the proper precision, its format must allow at least two decimal places.

Issuing a TABLE request against a Master File that specifies a currency code not listed in the
active currency database generates the following message:

(FOC1911) CURRENCY IN FILE DESCRIPTION NOT FOUND IN DATA

A syntax error or undefined field name in a currency conversion expression generates the
following message:

(FOC1912) ERROR IN PARSING CURRENCY STATEMENT
The following rates were in effect on December 31, 1998. Euroland countries as of that date are marked with an asterisk (*). Their rates are fixed and will not change; the rates for other countries can change over time:

<table>
<thead>
<tr>
<th>Country</th>
<th>Currency Code</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria*</td>
<td>ATS</td>
<td>13.7603</td>
</tr>
<tr>
<td>Belgium*</td>
<td>BEF</td>
<td>40.3399</td>
</tr>
<tr>
<td>Canada</td>
<td>CAD</td>
<td>1.7978</td>
</tr>
<tr>
<td>Denmark</td>
<td>DKK</td>
<td>7.46215</td>
</tr>
<tr>
<td>European Union</td>
<td>EUR</td>
<td>1</td>
</tr>
<tr>
<td>Finland*</td>
<td>FIM</td>
<td>5.94573</td>
</tr>
<tr>
<td>France*</td>
<td>FRF</td>
<td>6.55957</td>
</tr>
<tr>
<td>Germany*</td>
<td>DEM</td>
<td>1.95583</td>
</tr>
<tr>
<td>Greece</td>
<td>GRD</td>
<td>328.6</td>
</tr>
<tr>
<td>Ireland*</td>
<td>IEP</td>
<td>0.787564</td>
</tr>
<tr>
<td>Italy*</td>
<td>ITL</td>
<td>1936.27</td>
</tr>
<tr>
<td>Japan</td>
<td>JPY</td>
<td>133.149</td>
</tr>
<tr>
<td>Luxembourg*</td>
<td>LUF</td>
<td>40.3399</td>
</tr>
<tr>
<td>Netherlands*</td>
<td>NLG</td>
<td>2.20371</td>
</tr>
<tr>
<td>Norway</td>
<td>NOK</td>
<td>8.91039</td>
</tr>
<tr>
<td>Portugal*</td>
<td>PTE</td>
<td>200.482</td>
</tr>
<tr>
<td>Spain*</td>
<td>ESP</td>
<td>166.386</td>
</tr>
<tr>
<td>Sweden</td>
<td>SEK</td>
<td>9.52669</td>
</tr>
<tr>
<td>Switzerland</td>
<td>CHF</td>
<td>1.61093</td>
</tr>
<tr>
<td>UK</td>
<td>GBP</td>
<td>0.706739</td>
</tr>
<tr>
<td>USA</td>
<td>USD</td>
<td>1.17249</td>
</tr>
</tbody>
</table>
Example: Converting U.S. Dollars to Euros, French Francs, and Belgian Francs

Assume PRICE is denominated in U.S. dollars and ACTUAL is the name of a rate in the currency database. Using the currency conversion rates from Sample Currency Codes on page 8-12, the following FOCEXEC converts PRICE to euros, French francs, and Belgian francs:

```plaintext
/* CURRCODE IS THE CURRENCY DATABASE
   */
/* CURRDATA IS THE DATA SOURCE WITH CURRENCY-DENOMINATED FIELDS
   */

/* THE FOLLOWING FILEDEFS ARE FOR RUNNING UNDER CMS
CMS FILEDEF CURRCODE DISK CURRCODE TEXT A
CMS FILEDEF CURRDATA DISK CURRDATA TEXT A

/* THE FOLLOWING ALLOCATIONS ARE FOR RUNNING UNDER MVS
-DYNAM ALLOC FILE CURRCODE DA USER1.FOCEXEC.DATA(CURRCODE) SHR REU
-DYNAM ALLOC FILE CURRDATA DA USER1.FOCEXEC.DATA(CURRDATA) SHR REU

SET EUROFILE = CURRCODE

DEFINE FILE CURRDATA
PRICEEUR/P17.2 CURR EUR = EUR(PRICE, ACTUAL);
END

TABLE FILE CURRDATA
PRINT PRICE PRICEEUR AND COMPUTE
PRICEFRF/P17.2 CURR FRF = FRF(PRICE, ACTUAL);
PRICEBEF/P17.2 CURR BEF = BEF(PRICE, ACTUAL);
END

This request generates the following report:

<table>
<thead>
<tr>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRICE</td>
<td>PRICEEUR</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>5.00</td>
<td>4.26</td>
</tr>
<tr>
<td>6.00</td>
<td>5.12</td>
</tr>
<tr>
<td>40.00</td>
<td>34.12</td>
</tr>
<tr>
<td>10.00</td>
<td>8.53</td>
</tr>
</tbody>
</table>

Note: You cannot use the derived euro value PRICEEUR in a conversion from USD to BEF. PRICEEUR has two decimal places (P17.2), not three, as the triangulation rules require. Therefore, PRICEEUR yields the following inaccurate result (see PRICEBEF above) and is not valid as the intermediate value in a currency conversion that requires triangulation:

```plaintext
COMPUTE PRICENEW/P17.2 CURR BEF = BEF(PRICEEUR, ACTUAL);  
PRICENEW  
--------
171.85  
206.54  
1376.40 
344.10  
```
A Master Files and Diagrams

Topics:
- The EMPLOYEE Database
- The JOBFILE Database
- The EDUCFILE Database
- The SALES Database
- The PROD Database
- The CAR Database
- The LEDGER Database
- The FINANCE Database
- The REGION Database
- The COURSES Database
- The EMPDATA Database
- The EXPERSON Database
- The TRAINING Database
- The PAYHIST File
- The COMASTER File
- VideoTrk and Movies Databases
Master Files and Diagrams

You can create these files on your user ID by executing the FOCEXEC procedures specified below. These FOCEXECs were supplied for your installation with FOCUS. If they are not available to you or if they produce error messages, contact your systems administrator.

To create these files, first make sure you have read access to the Master Files, then perform the following:

- To create the EMPLOYEE, EDUCFILE, and JOBFILE databases, under CMS enter:
  
  EX EMPTEST

  Under MVS, enter:
  
  EX EMPTSO

  These FOCEXECs also test the databases by generating sample reports. If you are using Hot Screen, remember to press either Enter or the PF3 key after each report. If the EMPLOYEE, EDUCFILE, and JOBFILE databases already exist on your user ID, the FOCEXEC will replace the databases with new copies. This FOCEXEC assumes that the high-level qualifier for the FOCUS databases will be the same as the high-level qualifier for the MASTER PDS that was unloaded from the tape.

- To create the SALES and PROD databases, enter:
  
  EX SALES
  EX PROD

  This FOCEXEC will only create the SALES and PROD databases if they do not already exist; it will not replace a database. To replace a database, first erase it (CMS ERASE, TSO DELETE, or DYNAM DELETE), then execute GSTART.

- The CAR database is created automatically during the installation process.

- To create the LEDGER, FINANCE, REGION, COURSES, and EXPERSOEN databases, execute the appropriate FOCEXEC. For example, to create the LEDGER database, enter:
  
  EX LEDGER

- To create the EMPDATA and TRAINING databases, execute the LOADEMP and LOADTRAI FOCEXECs, respectively.

- The corresponding file for the PAYHIST Master File is called PAYHIST DATA. It is a sequential file and is allocated during the installation process.

- A corresponding FOCEXEC does not exist for the COMASTER database, because COMASTER is used for debugging other Master Files.

- To create the VideoTrk and Movies databases issue:
  
  EX LOADVTRK
The EMPLOYEE Database

The EMPLOYEE database contains data about a company’s employees. Its segments are:

- **EMPINFO**, which contains employee IDs, names, and positions.
- **FUNDTRAN**, which specifies employees’ direct deposit accounts. This segment is unique.
- **PAYINFO**, which contains the employee’s salary history.
- **ADDRESS**, which contains employees’ home and bank addresses.
- **SALINFO**, which contains data on employees’ monthly pay.
- **DEDUCT**, which contains data on monthly pay deductions.

The EMPLOYEE database also contains cross-referenced segments belonging to the JOBFILE and EDUCFILE files, described later in this appendix. The segments are:

- **JOBSEG** (from JOBFILE), which describes the job positions held by each employee.
- **SECSEG** (from JOBFILE), which lists the skills required by each position.
- **SKILLSEG** (from JOBFILE), which specifies the security clearance needed for each job position.
- **ATTNDSEG** (from EDUCFILE), which lists the dates that employees attended in-house courses.
- **COURSEG** (from EDUCFILE), which lists the courses that the employees attended.
### The EMPLOYEE Master File

**FILENAME**-EMPLOYEE, **SUFFIX**-FOC  
**SEGNAME**-EMPINFO, **SEGTYPE**-S1  
- **FILENAME**-EMP_ID,  **ALIAS**-EID,  **FORMAT**-A9  
- **FILENAME**-LAST_NAME,  **ALIAS**-LN,  **FORMAT**-A15  
- **FILENAME**-FIRST_NAME,  **ALIAS**-FN,  **FORMAT**-A10  
- **FILENAME**-HIRE_DATE,  **ALIAS**-HDT,  **FORMAT**-16YMD  
- **FILENAME**-DEPARTMENT,  **ALIAS**-DPT,  **FORMAT**-A10  
- **FILENAME**-CURR_SAL,  **ALIAS**-CSAL,  **FORMAT**-D12.2M  
- **FILENAME**-CURR_JOBCODE,  **ALIAS**-CJC,  **FORMAT**-A3  
- **FILENAME**-ED_HRS,  **ALIAS**-EJT,  **FORMAT**-F6.2  
- **SEGNAME**-FUNDTRAN,  **SEGTYPE**-U,  **PARENT**-EMPINFO  
- **FILENAME**-BANK_NAME,  **ALIAS**-BN,  **FORMAT**-A20  
- **FILENAME**-BANK_CODE,  **ALIAS**-BC,  **FORMAT**-I6S  
- **FILENAME**-BANK_ACCT,  **ALIAS**-BA,  **FORMAT**-I3S  
- **FILENAME**-EFFECT_DATE,  **ALIAS**-EDATE,  **FORMAT**-16YMD  
- **SEGNAME**-PAYINFO,  **SEGTYPE**-SHL,  **PARENT**-EMPINFO  
- **FILENAME**-DAT_INC,  **ALIAS**-DI,  **FORMAT**-16YMD  
- **FILENAME**-PCT_INC,  **ALIAS**-PI,  **FORMAT**-F6.2  
- **FILENAME**-SALARY,  **ALIAS**-SA,  **FORMAT**-D12.2M  
- **FILENAME**-JOBCODE,  **ALIAS**-JBC,  **FORMAT**-A3  
- **SEGNAME**-ADDRESS,  **SEGTYPE**-S1,  **PARENT**-EMPINFO  
- **FILENAME**-TYPE,  **ALIAS**-AT,  **FORMAT**-A4  
- **FILENAME**-ADDRESS_LM1,  **ALIAS**-LM1,  **FORMAT**-A20  
- **FILENAME**-ADDRESS_LM2,  **ALIAS**-LM2,  **FORMAT**-A20  
- **FILENAME**-ADDRESS_LM3,  **ALIAS**-LM3,  **FORMAT**-A20  
- **FILENAME**-PENUMBER,  **ALIAS**-PN,  **FORMAT**-19L  
- **SEGNAME**-SALINFO,  **SEGTYPE**-SHL,  **PARENT**-EMPINFO  
- **FILENAME**-PAY_DATE,  **ALIAS**-PD,  **FORMAT**-16YMD  
- **FILENAME**-GROSS,  **ALIAS**-GROSS,  **FORMAT**-D12.2M  
- **SEGNAME**-DEDUCT,  **SEGTYPE**-S1,  **PARENT**-SALINFO  
- **FILENAME**-DED_CODE,  **ALIAS**-DC,  **FORMAT**-A4  
- **FILENAME**-DED_AMT,  **ALIAS**-DA,  **FORMAT**-D12.2M  
- **SEGNAME**-JOBSITE,  **SEGTYPE**-KU,  **PARENT**-PAYINFO,  **CFFILE**-JOBFILE,  **CFKEY**-JOBCODE  
- **SEGNAME**-SECSEG,  **SEGTYPE**-KLU,  **PARENT**-JOBSITE,  **CFFILE**-JDFILE  
- **SEGNAME**-SKILLSSEG,  **SEGTYPE**-KL,  **PARENT**-JOBSITE,  **CFFILE**-JDFILE  
- **SEGNAME**-ATMSEG,  **SEGTYPE**-KN,  **PARENT**-EMPINFO,  **CFFILE**-EDUCFILE,  **CFKEY**-EMP_ID  
- **SEGNAME**-COSESEG,  **SEGTYPE**-KLU,  **PARENT**-ATMSEG,  **CFFILE**-EDUCFILE
The EMPLOYEE Structure Diagram

SECTION 01  STRUCTURE OF FOCUS  FILE EMPLOYEE ON 01/05/96 AT 14.02.35
  05 I ENGINFO  06 I KL
  01 I S1

  *SEX**CC**  :SKILLS
  *EMP_ID**  :SKILL_DESC
  *LAST_NAME**  :
  *FIRST_NAME**  :
  *HIRE_DATE**  :

**********JOBFILE**********

**********JOBFILE**********

SECTION 02
  02 I FUNDTRAN  03 I PAYINFO  07 I ADDRESS  08 I SALINFO  10 I ATTDSEG

  *BANK_NAME**  *DATE_INC**  TYPE**  *PAY_DATE**  DATE_ATTEND**
  *BANK_CODE**  *PCT_INC**  ADDRESS_LM1**  GROSS**  EMP_ID**
  *BANK_ACCOUNT**  *SALARY**  ADDRESS_LM2**  :
  *EFF_DATE**  *JOB_CODE**  ADDRESS_LM3**  :
  :

**********JOBFILE**********

SECTION 04
  04 I KU

  *JOB_CODE**
  *JOB_DESC**

**********JOBFILE**********

SECTION 09
  09 I S1

  *DED_CODE**  *COURSE_CODE**
  *DENT**  *COURSE_NAME**
  :
  :

**********EDUCFILE**********
The JOBFILE Database

The JOBFILE database contains information on a company's job positions. Its segments are:

- JOBSEG describes what each position is. The field JOBCODE in this segment is indexed.
- SKILLSEG lists the skills required by each position.
- SECSEG specifies the security clearance needed, if any. This segment is unique.

The JOBFILE Master File

FILENAME=JOBFILE, SUFFIX=POC  
SEGNAME=JOBSEG, SEGTYPE=S1  
FIELD=JOBCODE, ALIAS=JC, USAGE=A3, INDEX=1, S  
FIELD=JOB_DESC, ALIAS=JD, USAGE=A25, S  
SEGNAME=SKILLSEG, SEGTYPE=S1, PARENT=JOBSEG  
FIELD=SKILLS, ALIAS=, USAGE=A4, S  
FIELD=SKILL_DESC, ALIAS=SD, USAGE=A30, S  
SEGNAME=SECSEG, SEGTYPE=U, PARENT=JOBSEG  
FIELD=SEC_CLEAR, ALIAS=SC, USAGE=A6, S

The JOBFILE Structure Diagram

SECTION 01  STRUCTURE OF FOCUS  FILE JOBFILE  ON 01/05/96 AT 14:40:06

JOBSEG
01 S1

***************
<JOBCODE >>I
<JOB_DESC >>
= = =
= = =
= = =
= = =
= = =
= = =
= = =
= = =
= = =

***************

I

I

I

I

I

I

SECSEG  I  SKILLSEG
02  I  U  03  I  S1

***************
<SEC_CLEAR >>S
<SKILLS >>
= = =
= = =
= = =
= = =
= = =

***************

A-6  Information Builders
The EDUCFILE Database

The EDUCFILE database contains data on a company’s in-house courses. Its segments are:

- COURSEG contains data on each course.
- ATTNDSEG specifies which employees attended the courses. Both fields in the segment are key fields. The field EMP_ID in this segment is indexed.

The EDUCFILE Master File

```plaintext
FILENAME=EDUCFILE ,SUFFIX=FOC
SEGNAMES=COURSEG ,SEGTYPE=S1
FIELD=COURSE_CODE ,ALIAS=CC ,USAGE=A6 ,S
FIELD=COURSE_NAME ,ALIAS=CD ,USAGE=A30 ,S
SEGNAMES=ATTNDSEG ,SEGTYPE=SH2 ,PARENT=COURSEG
FIELD=DATE_ATTEND ,ALIAS=DA ,USAGE=A1M ,S
FIELD=EMP_ID ,ALIAS=EI ,USAGE=A3 ,INDEX=1 ,S
```

The EDUCFILE Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS FILE EDUCFILE ON 01/05/96 AT 14:45:44

COURSEG
  01 S1
    ***************
    *COURSE_CODE**
    *COURSE_NAME**
    *  **
    *  **
    *  **
    ***************
    ***************
    I I I
    I I
    I ATTNSEG

02 SH2
    ***************
    *DATE_ATTEND**
    *EMP_ID  **I
    *  **
    *  **
    *  **
    ***************
```

Developing Applications
The SALES Database

The SALES database records sales data for a dairy company (or a store chain). Its segments are:

- **STOR SEG** lists the stores buying the products.
- **DAT SEG** contains the dates of inventory.
- **PRODUCT** contains sales data for each product on each date. Note the following about fields in this segment:
  - The **PROD_CODE** field is indexed.
  - The **RETURNS** and **DAMAGED** fields have the **MISSING=ON** attribute.

The SALES Master File

```plaintext
FILENAME=KSALES, SUFFIX=FOC,

SEGNAME=STOR_SEG, SECTYPE=S1,
  FIELDMNAME=STORE_CODE, ALIAS=SM0, FORMAT=A3, $
  FIELDMNAME=CITY, ALIAS=CTY, FORMAT=A15, $
  FIELDMNAME=AREA, ALIAS=LOC, FORMAT=A1, $

SEGNAME=DATE_SEG, PARENT=STOR_SEG, SECTYPE=SM1,
  FIELDMNAME=DATE, ALIAS=DATE, FORMAT=ADMD, $

SEGNAME=PRODUCT, PARENT=DATE_SEG, SECTYPE=S1,
  FIELDMNAME=PROD_CODE, ALIAS=PCODE, FORMAT=A3, FIELDMTYPE=I, $
  FIELDMNAME=UNIT_SOLD, ALIAS=SLD, FORMAT=I5, $
  FIELDMNAME=RETAIL_PRICE, ALIAS=RP, FORMAT=DF2M, $
  FIELDMNAME=DELIVER_AMT, ALIAS=SHIP, FORMAT=I5, $
  FIELDMNAME=OPENING_AMT, ALIAS=INV, FORMAT=I5, $
  FIELDMNAME=RETURNS, ALIAS=RTN, FORMAT=I3, MISSING=ON, $
  FIELDMNAME=DAMAGED, ALIAS=BAD, FORMAT=I3, MISSING=ON, $
```
The SALES Database

The SALES Structure Diagram

SECTION 01
STRUCTURE OF FOCUS  FILE SALES  ON 01/05/96 AT 14:50:28

STOR_SEG
01  S1
**************
»STORE_CODE  **
»CITY  **
»AREA  **
»  **
»  **
**************
**************
  I
  I
  I
  I
  I
I_DATE_SEG
02  S1
**************
»DATE  **
»  **
»  **
»  **
»  **
**************
**************
  I
  I
  I
  I
  I
I_PRODUCT
03  S1
**************
»PROD_CODE  **I
»UNIT_SOLD  **
»RETAIL_PRICE**
»DELIVER_AMT**
»  **
**************
Master Files and Diagrams

The PROD Database

The PROD database lists products sold by a dairy company. It consists of one segment, PRODUCT. The field PROD_CODE is indexed.

The PROD Master File

FILE=KPROD, SUFFIX=FOC,

SEGMENT=PRODUCT, SEGTYPE=S1,
   FIELDNAME=PROD_CODE, ALIAS=PCODE, FORMAT=A3, FIELDTYPE=I, $
   FIELDNAME=PROD_NAME, ALIAS=ITEM, FORMAT=A15, $
   FIELDNAME=PACKAGE, ALIAS=SIZE, FORMAT=A12, $
   FIELDNAME=UNIT_COST, ALIAS=COST, FORMAT=05.2M, $

The PROD Structure Diagram

SECTION 01

STRUCTURE OF FOCUS  FILE PROD  ON 01/05/94 AT 14:57:38

PRODUCT

01 S1

============

*PROD_CODE  <=|
*PROD_NAME  <=
*PACKAGE    <=
*UNIT_COST  <=
*<=

============
The CAR Database

The CAR database contains specifications and sales information for rare cars. Its segments are:

- ORIGIN lists the country that manufactures the car. The field COUNTRY is indexed.
- COMP contains the car name.
- CARREC contains the car model.
- BODY lists the body type, seats, dealer and retail costs, and units sold.
- SPECS lists car specifications. This segment is unique.
- WARANT lists the type of warranty.
- EQUIP lists standard equipment.

The aliases in the CAR Master File are specified without the ALIAS keyword.

The CAR Master File

```
FILENAME=CAR,SUFFIX=FNC
SEGNAME=ORIGIN,SECTYPE=S1
    FILENAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SECTYPE=S1,PARENT=ORIGIN
    FILENAME=CAR,CARS,A16,$
SEGNAME=CARREC,SECTYPE=S1,PARENT=COMP
    FILENAME=MODEL,MODEL,A24,$
SEGNAME=BODY,SECTYPE=S1,PARENT=CARREC
    FILENAME=BODYTYPE,TYPE,A12,$
    FILENAME=SEATS,SEAT,13,$
    FILENAME=DEALER_COST,DCOST,D7,$
    FILENAME=RETAIL_COST,RCOST,D7,$
    FILENAME=SALES,UNITS,16,$
SEGNAME=SPECS,SECTYPE=U,PARENT=BODY
    FILENAME=LENGTH,LEN,D5,$
    FILENAME=WIDTH,WIDTH,D5,$
    FILENAME=HEIGHT,HEIGHT,D5,$
    FILENAME=WEIGHT,WEIGHT,D6,$
    FILENAME=WHEELBASE,BASE,D6.1,$
    FILENAME=FUEL_CAP,FUEL,D6.1,$
    FILENAME=MPH,POWER,D6,$
    FILENAME=MPH,HPM,D6.5,$
    FILENAME=MILES,MILES,6,$
    FILENAME=ACCEL.SECONDS,D6,$
SEGNAME=WARANT,SECTYPE=S1,PARENT=COMP
    FILENAME=WARRANTY,WARR,A40,$
SEGNAME=EQUIP,SECTYPE=S1,PARENT=COMP
    FILENAME=STANDARD,EQUIP,A40,$
```
The LEDGER Database

The LEDGER database lists accounting information. It consists of one segment, TOP. This database is specified primarily for EMR examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The LEDGER Master File

```plaintext
FILENAME-LEDGER. SUFFIX=FOC.$
SEGNAME=TOP, SEGTYPE=SZ,$
FIELDNAME=YEAR , , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $'
FIELDNAME=AMOUNT , , FORMAT=15C,$
```

The LEDGER Structure Diagram

```
SECTION 01
STRUCTURE OF FOCUS   FILE LEDGER ON 01/05/96 AT 15.07.56

TOP
  01   SZ
  ***************
  $YEAR $*
  $ACCOUNT $*
  $AMOUNT $*
  $*
  $*
  ***************
  ***************
```
Master Files and Diagrams

The FINANCE Database

The FINANCE database contains financial information for balance sheets. It consists of one segment, TOP. This database is specified primarily for EMR examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The FINANCE Master File

```
FILENAME-FINANCE, SUFFIX-FOC,$
SEGNAME=TOP, SEGTYPE=S2,$
FIELDNAME=YEAR , , FORMAT=A4, $
FIELDNAME=ACCOUNT, , FORMAT=A4, $
FIELDNAME=AMOUNT , , FORMAT=D12C,$
```

The FINANCE Structure Diagram

```
SECTION 01
 structure of focus

FILE FINANCE ON 01/05/96 AT 15.17.08

TOP

S2

**************
+YEAR  **
+ACCOUNT  **
+AMOUNT  **
*  **
*  **
**************
**************
```
The REGION Database

The REGION database lists account information for the east and west regions of the country. It consists of one segment, TOP. This database is specified primarily for EMR examples. Aliases do not exist for the fields in this Master File, and the commas act as placeholders.

The REGION Master File

```
FILENAME=REGION, SUFFIX=FDC,$
SEGNAME=TOP, SEGTYPE=S1,$
FIELDNAME=ACCOUNT , , FORMAT=A4, $
FIELDNAME=E_ACTUAL , , FORMAT=15C,$
FIELDNAME=E_BUDGET, , FORMAT=15C,$
FIELDNAME=W_ACTUAL , , FORMAT=15C,$
FIELDNAME=W_BUDGET, , FORMAT=15C,$
```

The REGION Structure Diagram

```
SECTION 01  STRUCTURE OF FOCUS  FILE REGION  ON 01/05/96 AT 15.18.48

  01   S1
   **************
  **ACCOUNT    **
  **E_ACTUAL   **
  **E_BUDGET  **
  **W_ACTUAL  **
  *          **
***************
***************
```
The COURSES Database

The COURSES database describes education courses. It consists of one segment, CRSESEG1.
The field DESCRIPTION has a format of TEXT (TX).

The COURSES Master File

```
FILENAME=COURSES, SUFFIX=FOC, 
SEGNAME=CRSESEG1, SEGTYPE=S1, 
  FIELDNAME=COURSE_CODE, ALIAS=CC, FORMAT=A6, FIELDTYPE=I, 
  FIELDNAME=COURSE_NAME, ALIAS=CN, FORMAT=A30, 
  FIELDNAME=DURATION, ALIAS=DAYS, FORMAT=I3, 
  FIELDNAME=DESCRIPTION, ALIAS=DESC, FORMAT=TX50, 
``` 

The COURSES Structure Diagram

```
SECTION 01

FILE COURSES ON 01/05/94 AT 15.29.59

CRSESEG1

01 S1

***************
*COURSE_CODE ??I
*COURSE_NAME ??
*DURATION ??
*DESCRIPTION ??I
* ??
***************
```
The EMPDATA Database

The EMPDATA database contains organizational data about a company’s employees. It consists of one segment, EMPDATA. Note the following:

- The PIN field is indexed.
- The AREA field is a temporary one.

The EMPDATA Master File

FILENAME=EMPDATA, SUFFIX=FDC
SEGNMAME=EMPDATA, SECTYPE=S1

FIELDNAME=PIN, ALIAS=ID, FORMAT=A9, INDEX=I, $  
FIELDNAME-LASTNAME, ALIAS=LN, FORMAT=A15, $  
FIELDNAME-FIRSTNAME, ALIAS=FN, FORMAT=A10, $  
FIELDNAME-MIDINITIAL, ALIAS=M1, FORMAT=A1, $  
FIELDNAME-DEPT, ALIAS=DEPT, FORMAT=A20, $  
FIELDNAME-JOBCLASS, ALIAS=CLAS, FORMAT=A9, $  
FIELDNAME-TITLE, ALIAS=TITLE, FORMAT=A20, $  
FIELDNAME-SALARY, ALIAS=CSAL, FORMAT=D12.2M, $  
FIELDNAME-HIREDATE, ALIAS=HDATE, FORMAT=YMD, $  

$  
DEFINE AREA=A13=DECODE DEPT (ONE 'NORTH EASTERN' SE 'SOUTH EASTERN'  
OE 'CENTRAL' WE 'WESTERN' CP 'CORPORATE' ELSE 'INVALID AREA');$  

The EMPDATA Structure Diagram

SECTION 01  STRUCTURE OF FOCUS  FILE EMPDATA  ON 01/05/96 AT 14.49.09

EMPDATA

01 S1

***************

*PIN   **1   
*LASTNAME  **  
*FIRSTNAME  **  
*MIDINITIAL   **  
*       **   

***************

***************
The EXPERSON Database

The EXPERSON database contains personal data about individual employees. It consists of one segment, ONESEG.

The EXPERSON Master File

```
FILE=EXPERSON , SUFFIX=FOC
SEGMENT=ONESEG ,$

FIELDNAME=SUC_SEC_NO , ALIAS=SSN , USAGE=A9 ,$
FIELDNAME-FIRST_NAME , ALIAS=FM , USAGE=A9 ,$
FIELDNAME-LAST_NAME , ALIAS=LM , USAGE=A10 ,$
FIELDNAME-AGE , ALIAS=YEARS , USAGE=A12 ,$
FIELDNAME-SEX , ALIAS= , USAGE=A1 ,$
FIELDNAME=MARITAL_STAT , ALIAS=MS , USAGE=A1 ,$
FIELDNAME-NO_DEF , ALIAS=NOP , USAGE=A13 ,$
FIELDNAME-DEGREE , ALIAS= , USAGE=A3 ,$
FIELDNAME-MO_CARS , ALIAS=CARS , USAGE=A13 ,$
FIELDNAME-ADDRESS , ALIAS= , USAGE=A14 ,$
FIELDNAME-CITY , ALIAS= , USAGE=A10 ,$
FIELDNAME-WAGE , ALIAS=PAY , USAGE=D10.25 ,$
FIELDNAME-CATEGORY , ALIAS=STATUS , USAGE=A1 ,$
FIELDNAME-SKILL_CODE , ALIAS=SKILLS , USAGE=A5 ,$
FIELDNAME-DEPT_CODE , ALIAS=WHERE , USAGE=A4 ,$
FIELDNAME-TEL_EXT , ALIAS=EXT , USAGE=A14 ,$
FIELDNAME-DATE_EMP , ALIAS=BASE_DATE , USAGE=A16,mid ,$
FIELDNAME-MULTIPLIER , ALIAS=RATIO , USAGE=D5.3 ,$
```

The EXPERSON Structure Diagram

```
SECTION 01

STRUCTURE OF FOCUS

FILE EXPERSON ON 01/05/96 AT 14.50.58

ONESEG

01 S1

************************************************************************
* SUC_SEC_NO **
* FIRST_NAME **
* LAST_NAME **
* AGE **
* **
************************************************************************
```
The TRAINING Database

The TRAINING database contains training course data for employees. It consists of one segment, TRAINING. Note the following:

- The PIN field is indexed.
- The EXPENSES, GRADE, and LOCATION fields have the MISSING=ON attribute.

The TRAINING Master File

FILENAME=TRAINING, SUFFIX=FOC
SEGNAME=TRAINING, SECTYPE=SN3

FILENAME=PIN, ALIAS-ID, FORMAT=AI, INDEX=I, $
FILENAME=COURSESTART, ALIAS-CSTART, FORMAT=XMD, $
FILENAME=COURSECODE, ALIAS-CCOD, FORMAT=A7, $
FILENAME=EXPENSES, ALIAS-COST, FORMAT=DA.2, MISSING=ON,$
FILENAME=GRADE, ALIAS-GRAD, FORMAT=A2, MISSING=ON,$
FILENAME=LOCATION, ALIAS-LOC, FORMAT=A6, MISSING=ON,$

The TRAINING Structure Diagram

SECTION 01

STRUCTURE OF FOCUS

FILE TRAINING ON 12/12/94 AT 14:51.28

TRAINING

01 SN3

***************
*PIN  **I*
*COURSESTART **
*COURSECODE **
*EXPENSES **
* $
***************

***************
The PAYHIST File

The PAYHIST database contains the employees’ salary history. It consists of one segment, PAYSEG. The SUFFIX attribute indicates that the data file is a fixed-format sequential file.

The PAYHIST Master File

FILENAME=PAYHIST, SUFFIX=FIX
SEGMENT=PAYSEG,$
   FIELDDNME=SOC_SEC_NO, ALIAS=SSN, USAGE=AG, ACTUAL=AG,$
   FIELDDNME=DATE_OF_IN, ALIAS=INCDATE, USAGE=BD, ACTUAL=BD,$
   FIELDDNME=ANTF_OF_INC, ALIAS=RAISE, USAGE=CD, ACTUAL=CD,$
   FIELDDNME=PCI_INC, ALIAS=PIC, USAGE=CD, ACTUAL=CD,$
   FIELDDNME=NEW_SAL, ALIAS=CURR_SAL, USAGE=DD, ACTUAL=DD,$
   FIELDDNME=FULL, ALIAS=, USAGE=EF, ACTUAL=EF,$

The PAYHIST Structure Diagram

SECTION 01  STRUCTURE OF PAYHIST FILE PAYHIST ON 01/05/96 AT 14:51:53

**PAYSEG**

01  S1

**************
*SOC_SEC_NO  **
*DATE_OF_IN  **
*ANT_OF_INC  **
*PCI_INC  **
*  **
**************
**************
The COMASTER File

The COMASTER file is used to display the file structure and contents of each segment in a data source. Since COMASTER is used for debugging other Master Files, a corresponding FOCEXEC does not exist for the COMASTER file. Its segments are:

- FILEID lists file information.
- RECID lists segment information.
- FIELDID lists field information.
- DEFREC lists a description record.
- PASSREC lists read/write access.
- CRSEG lists cross-reference information for segments.
- ACCSEG lists DBA information.
### Master Files and Diagrams

#### The COMASTER Master File

```plaintext
FILE=COMASTER, SUFFIX=CM.

SEGNAME=FILEID
FIELDNAME=FILENAME , FILE , A8 , $
FIELDNAME=SUFFIX , SUFFIX , A8 , $

SEGNAME=RECID
FIELDNAME=SEGNAME , SEGMENT , A8 , $
FIELDNAME=SEGTYPE , SEGTYPE , A4 , $
FIELDNAME=SEGSIZE , SEGSIZE , A14 , A4,$
FIELDNAME=PARENT , PARENT , A8 , $
FIELDNAME=CRRKEY , CRRKEY , A66 , $

SEGNAME=FIELDID
FIELDNAME=FILENAME , FIELD , A66 , $
FIELDNAME=FLIAS , SYNONYM , A66 , $
FIELDNAME=FORMAT , USAGE , A8 , $
FIELDNAME=ACTUAL , ACTUAL , A8 , $
FIELDNAME=AUTHORITY , AUTHCODE , A8 , $
FIELDNAME=FIELDTYPE , INDEX , A8 , $
FIELDNAME=TITLE , TITLE , A64 , $
FIELDNAME=HELPMESSAGE , MESSAGE , A256 , $
FIELDNAME=MISSING , MISSING , A4 , $
FIELDNAME=ACCEPTS , ACCEPTABLE , A255 , $
FIELDNAME=RESERVED , RESERVED , A44 , $

SEGNAME=DEFREC
FIELDNAME=DEFINITION , DESCRIPTION , A44 , $

SEGNAME=PASSREC,PARENT=FILEID
FIELDNAME=READ WRITE , RW , A32 , $

SEGNAME=CSSEG,PARENT=RECID
FIELDNAME=CSFILENAME , CSFILE , A8 , $
FIELDNAME=CSSEGNAME , CSSEGMENT , A8 , $
FIELDNAME=CRYPT , CRYPT , A4 , $

SEGNAME=ACSEG,PARENT=DEFREC
FIELDNAME=DB0 , DBA , A8 , $
FIELDNAME=PROFILE , , A8 , $
FIELDNAME=USER , PASS , A8 , $
FIELDNAME=ACCESS , ACCESS , A8 , $
FIELDNAME=RESTRICT , RESTRICT , A8 , $
FIELDNAME=NAME , NAME , A66 , $
FIELDNAME=VALUE , VALUE , A66 , $
```

A-22

Information Builders
The COMASTER File

The COMASTER Structure Diagram

SECTION 01

STRUCTURE OF EXTERNAL FILE COMASTER ON 12-12-94 AT 14:53:38

FILEID
01 S1

***************
*FILENAME ==
*FILE SUFFIX ==
* ==
* ==
* ==

***************

I

I RECID I PASSREC
02 IN 07 IN

***************

*SEGNAME == READ/WRITE ==
*SEGTYPE == x ==
*SEGSIZE == x ==
*SEGKEY == x ==
* == x ==

***************

I

I

FIELID I SEQSEG
03 IN 06 IN

***************

*FILENAME == CHFILENAME ==
*NAME == CASEGNAME ==
*NAMES == INCRIPT ==
*ACTUAL == x ==
* == x ==

***************

I

I

I REPREC
04 IN

***************

*DEFINITION ==
* ==
* ==
* ==

***************

I

I

I SEQSEG
05 IN

***************

*DBA ==
*DBFILE ==
*DBER ==
*ACCESS ==
* ==

***************

Developing Applications
Master Files and Diagrams

VideoTrk and Movies Databases

The VideoTrk database tracks customer, rental, and purchase information for a video rental business. It can be joined to the Movies database. VideoTrk and Movies are used in examples that illustrate the use of the Maintain facility.

VideoTrk Master

FILENAME=VIDEOTRK, SUFFIX=FOC
SEGNAME=CUST, SEGTYPE=S1
  FIELDDNAME=CUSTID, ALIAS=CIN, FORMAT=A4, $
  FIELDDNAME=LASTNAME, ALIAS=LN, FORMAT=A15, $
  FIELDDNAME=FIRSTNAME, ALIAS=FN, FORMAT=A10, $
  FIELDDNAME=EXPDATE, ALIAS=EXDAT, FORMAT=YMD, $
  FIELDDNAME=PHONE, ALIAS=TEL, FORMAT=A10, $
  FIELDDNAME=STREET, ALIAS=STR, FORMAT=A20, $
  FIELDDNAME=CITY, ALIAS=CITY, FORMAT=A20, $
  FIELDDNAME=STATE, ALIAS=PROV, FORMAT=A4, $
  FIELDDNAME=ZIP, ALIAS=POSTAL_CODE, FORMAT=A9, $
SEGNAME=TRANSDAT, SEGTYPE=SH1, PARENT=CUST
  FIELDDNAME=TRANSDATE, ALIAS=OUTDATE, FORMAT=YMD, $
SEGNAME=SALES, SEGTYPE=S2, PARENT=TRANSDAT
  FIELDDNAME=PRODCODE, ALIAS=PCOD, FORMAT=A6, $
  FIELDDNAME=TRANSCODE, ALIAS=TCOD, FORMAT=I3, $
  FIELDDNAME=QUANTITY, ALIAS=NO, FORMAT=I3S, $
  FIELDDNAME=TRANSTOT, ALIAS=TTOT, FORMAT=F7.2S, $
SEGNAME=RENTALS, SEGTYPE=S2, PARENT=TRANSDAT
  FIELDDNAME=MOVIECODE, ALIAS=MCOD, FORMAT=A6, INDEX=I, $
  FIELDDNAME=COPY, ALIAS=COPI, FORMAT=I2, $
  FIELDDNAME=RETURNDATE, ALIAS=INDATE, FORMAT=YMD, $
  FIELDDNAME=FEE, ALIAS=FEE, FORMAT=F5.2S, $

Movies Master

FILENAME=Movies, SUFFIX=FOC
SEGNAME=MOVINFO, SEGTYPE=S1
  FIELDDNAME=MOVIECODE, ALIAS=MCOD, FORMAT=A6, INDEX=I, $
  FIELDDNAME=TITLE, ALIAS=MTL, FORMAT=A39, $
  FIELDDNAME=CATEGORY, ALIAS=CLASS, FORMAT=A8, $
  FIELDDNAME=DIRECTOR, ALIAS=DIR, FORMAT=A17, $
  FIELDDNAME=RATING, ALIAS=RTG, FORMAT=A4, $
  FIELDDNAME=RELDATE, ALIAS=RDAT, FORMAT=YMD, $
  FIELDDNAME=WHOLESALEPR, ALIAS=WPRC, FORMAT=F6.2, $
  FIELDDNAME=LISTPR, ALIAS=LPRC, FORMAT=F6.2, $
  FIELDDNAME=COPYS, ALIAS=NOC, FORMAT=I3, $

A-24

Information Builders
VideoTrk Structure Diagram

SECTION 01

CUSTOMER STRUCTURE OF FOCUS FILE VIDEOTRK ON 05/21/99 AT 12.25.19

CUST
01  S1
*-------------
*CUSTID **
*LASTNAME **
*FIRSTNAME **
*EXPDATE **
* **
*-------------

I
I
I
I TRANSDAT
02  S1H1
*-------------
*TRANSDATE **
* **
* **
* **
* **********

TRANSACTION
03  S2
*-------------
*PRODCODE **
*MOVIECODE **I
*TRANSCODE **
*COPY **
*QUANTITY **
*RETURNDATE **
*TRANSTOT **
*FEE **
* **
* **********

SALES RENTALS
I
I
I SALES I RENTALS
03  S2 04  S2
*-------------
*-------------
Movies Structure Diagram

SECTION 01
STRUCTURE OF FOCUS  FILE MOVIES  ON 05/21/99 AT 12.26.05

MOVINFO
01   S1
**************
*MOVIECODE   **I
*TITLE       **
*CATEGORY    **
*DIRECTOR    **
*             **
**************
**************
B Error Messages

If you need to see the text or explanation for any error message, you can display it online in your FOCUS session or find it in a standard FOCUS ERRORS file. All of the FOCUS error messages are stored in eight system ERRORS files:

- For CMS, these files are:
  - FOT004 ERRORS
  - FOG004 ERRORS
  - FOM004 ERRORS
  - FOS004 ERRORS
  - FOA004 ERRORS
  - FSQLXLT ERRORS
  - FOCSTY ERRORS
  - FOB004 ERRORS

- For MVS, these files are the following members in the ERRORS PDS:
  - FOT004
  - FOG004
  - FOM004
  - FOS004
  - FOA004
  - FSQLXLT
  - FOCSTY
  - FOB004

To display a message online, issue the following query command at the FOCUS command level

? n

where n is the message number.
Error Messages

The message number and text will display along with a detailed explanation of the message (if one exists). For example, issuing the following command:

? 210

displays the following

(FOC210) THE DATA VALUE HAS A FORMAT ERROR:

An alphabetic character has been found where all numerical digits are required.
Index

Symbols

-" ", 2-107
& (local variables), 2-49
&& (global variables), 2-50
&ACCEPTS, 2-57
&BASEIO, 2-57
&CHNGD, 2-57
&CURSOR, 2-59
&CURSORAT, 2-59
&DATE, 2-51, 2-110
&DATEfmt, 1-30
&DELTd, 2-57
&DMY, 2-51, 2-110
&DMYY, 1-30, 2-51, 2-56
&DUPLs, 2-57
&ECHO, 2-40, 2-59
&FOCCPU, 2-51, 2-110
&FOCDISORG, 2-57, 6-9
&FOCERRNUM, 2-57
&FOCEXITRM, 2-51, 2-110
&FOCFIELDNAME, 2-52, 2-110
&FOCFEXEC, 2-52, 2-110
&FOCINCLUDE, 2-52, 2-110
&FOCMODE, 2-51, 2-111
&FOCPRINT, 2-51, 2-111
&FOCPUTLVL, 2-52, 2-111
&FOCQUALCHAR, 2-53, 2-111
&FOCREL, 2-53, 2-110
&FOCSBORDER, 2-53, 2-111
&FOCSYSTYP, 2-53
&FOCTMPDSK, 2-53, 2-111
&FOCTRMSD, 2-53, 2-111
&FOCTRMSW, 2-53, 2-111
&FOCTRTY, 2-53, 2-111
&FOCTTIME, 2-53, 2-111
&FOCVTIME, 2-53, 2-112
&FORMAT, 2-57
&HIPERFOCUS, 2-53, 2-112
&INPUT, 2-57
&INVALID, 2-57
&IORETURN, 2-54, 2-112
&LINES, 2-57
&MDY, 2-54, 2-112
&MDYY, 1-30, 2-54, 2-56
&NOMATCH, 2-57
&PFKEY, 2-59
&QUIT, 2-42, 2-59
&READs, 2-58
&RECORDs, 2-58
&REJECTs, 2-58
&RETCODE, 2-54, 2-112
Query command tests, 2-23
&STACK, 2-40, 2-59
&TOD, 2-54, 2-112
&TRANS, 2-58
&WINDOWNAME, 2-59
&WINDOWVALUE, 2-59
Index

&YMD, 2-54
&YYMD, 1-30, 2-54, 2-56
*, 2-78
*, 2-7
.EXIST operator, 2-20
.LENGTH operator, 2-21
.TYPE operator, 2-22
?, 2-79
? &., 2-48
? &&, 2-50
? && query command, 6-18
? DEFINE query command, 6-3
? FDT query command, 6-5
? FILE query command, 6-7
? FILEDEF, 2-23
? HOLD query command, 6-9
? JOIN query command, 6-10
? LANG query command, 6-11
? LET query command, 5-9, 6-11
? LOAD, 7-5
? n query command, 6-12
? PTF, 6-13
? SET [ALL]
  With sliding window, 1-11
? SET ALL query command, 6-14
? SET GRAPH, 6-15
? SET GRAPH, 6-16
? SET parameter values, 2-48
? SET query command, 6-14
? STAT, 2-58, 6-16
? SU, 2-23

A

Absolute File Integrity, 6-15
  Turning on and off, 4-35
ACCBLN, 4-2
ACCEPT
  Controlling defaults, 4-2
Access Files
  Loading into memory, 7-3
  Types of files, 7-3
Alias
  SET FIELDNAME, 4-15
ALL
  Parameter, 4-2
Allocating
  Disk space, 4-38
Alphanumeric
  Variable format, 2-73
Alternate file views
  AUTOPATH parameter, 4-4
Amper variables, 2-5, 2-46
AS
  Preserving field names, 4-3
ASNAMES parameter, 4-3
AUTOINDEX parameter, 4-4
Automatic prompting, 2-72, 2-73
  Variable values, 2-47
AUTOPATH parameter, 4-4
AUTOSTRATEGY parameter, 4-5
Base dates
  Displaying in FOCUS reports, 4-10
BINS parameter, 4-5
BLKCALC, 4-5
Block sizes
  System-determined, 4-5
BORDERS, 4-34
Borders in full-screen mode, 2-53, 2-111
BOTTOMMARGIN parameter, 4-7
Branching
  Conditional with -IF, 2-17
  In procedures, 2-23
  Testing values, 2-23
  Unconditional with -GOTO, 2-16
Buffers
  SET BINS parameter, 4-5
BY
  TABLE
    Hot Screen facility, 4-6
BYPANEL, 4-6

Cache memory, 4-7
  Number of reads performed, 6-18
CACHE parameter, 4-7
CAR sample database, A-11
CARTESIAN parameter, 4-8
CDN parameter, 4-9
Century. See DEFCENT
CHECK
  Diagrams, A-2
  With sliding window, 1-9
CLEAR
  LET, 5-11

CMS
  TEMPDISK, 4-38
-CMS RUN, 2-82
  Subroutines, 2-33
Columns
  Spacing
    SET SPACES parameter, 4-36
COMASTER, A-21
Command statistics, 6-16, 6-18
Commands
  Processor, 2-4
Comments
  In Dialogue Manager, 2-78
  In procedures, 2-7
COMPILE, 7-1, 7-4, 7-5, 7-6
  RUN, 7-6
COMPUTE
  And currency conversion, 8-9
  LET and, 5-7
  SET, 4-10
  With sliding window, 1-3, 1-24
Computing variable values
  -SET, 2-27
Continental Decimal Notation (CDN), 4-9
Control statements, 2-10
  '.', ',', 2-107
-CMS RUN, 2-82
-CRTCLEAR, 2-83
-CRTFORM, 2-84
-DEFAULTS, 2-85
  Initializing a loop, 2-28
-INCLUDE, 2-91
-MVS RUN, 2-32, 2-93
-PASS, 2-94
-PROMPT, 2-95
-QUIT, 2-96
-QUIT FOCUS, 2-15, 2-96
-READ, 2-69, 2-97
-TSO RUN, 2-102
-TYPE, 2-103
-WINDOW, 2-104, 2-105
Index

Control statements (continued)
-WRITE, 2-106

Conversion
-Of currencies, 8-1
-Rule for cross-century dates, 1-2

COURSES, A-16

Cross-century dates. See Sliding window
-CRTCLREAR, 2-83
-CRTFORM
- With sliding window, 1-5
-CRTFORM
- Cursor position values, 2-59
- Dialogue Manager, 2-84
- Full-screen data entry, 2-46, 2-47, 2-72
- PF key values, 2-47, 2-59

CURR, 8-6

Currency
- Conversion, 8-1
- Processing, 8-9
- Conversion calculations, 8-11
- Database for conversions, 8-3
- Euro support, 8-1
- Field
- Identifying, 8-5

CURRENCY_ID, 8-4

D

Databases
- Administration, 4-29
- Security
- SET PASS parameter, 4-29
- Statistics
- ? FILE query command, 6-7

Dates
- Base
- Displaying in FOCUS reports, 4-10
- Displaying formats, 1-29
- Formats
- Validation, 1-6
- With sliding window, 1-4
- Setting in Dialogue Manager, 2-31

Dates (continued)
- Time stamp, 1-30
- Validation, 1-6
- Variable, 2-51, 2-110
- Displaying without separators, 2-56

DATETIME, 4-11

DBCS support, 4-22, 6-11
- SHIFT parameter, 4-35

Debugging
- Procedures, 2-40, 2-41

DECODE
- Procedure variables, 2-29, 2-30

Default century. See DEFCENT
- DEFCENT, 1-2, 1-6
- In Master Files, 1-12
- Querying
- Current global value, 1-11
- Value in Master Files, 1-30
- With COMPUTE, 1-3, 1-24, 1-26
- With DEFINE, 1-3, 1-19

DEFINE
- And currency conversion, 8-9
- LET and, 5-7
- Listing current fields, 6-3
- With sliding window, 1-3, 1-5, 1-19
- In Master Files, 1-22

Dialogue Manager, 2-1
- .EVAL operator, 2-60
- Amper variables, 2-5
- Branching and testing, 2-23
- Calling subroutines, 2-31, 2-33

Commands
- .*, 2-78
- .?, 2-79
- CMS, 2-81
- DEFAULTS, 2-67
- EXIT, 2-12, 2-87
- GOTO, 2-16, 2-88
- IF, 2-17, 2-90
- label, 2-92
- REPEAT, 2-24, 2-98
Dialogue Manager

Commands (continued)
- RUN, 2-11, 2-100
- SET, 2-68, 2-101

Concepts of execution, 2-5
Control statements, 2-8, 2-107
Creating procedures, 2-6
Debugging procedures, 2-40, 2-41
Executing procedures, 2-15
Executing Windows, 2-72, 2-104, 2-105

Full-screen menus, 2-72

General rules

Procedures, 2-6
Summary, 2-108, 2-112
Incorporating multiple procedures, 2-35

Loops, 2-28
Menus and windows, 2-72
Open-ended procedures, 2-39
Password in procedures, 2-7
PROFILE, 2-34
Profiles, 2-34
Prompting, 2-46, 2-70, 2-71, 2-74

Direct with -PROMPT, 2-95
Implied, 2-47, 2-72, 2-73
Variable values, 2-70
Querying parameter settings, 2-48
- READ, 2-69, 2-97
Statistical variables, 2-5, 2-19
Supplying variable values, 2-65, 2-67, 2-70,
2-71, 2-73, 2-76

Direct prompting, 2-70
Implied prompting, 2-72
System variables, 2-5, 2-19
Variables

Concatenation, 2-62
In procedures, 2-46
Verifying variable input values, 2-48, 2-73, 2-76
- WRITE, 2-42, 2-106

DISPLAY, 4-11

Displaying

Output window in TOE, 6-19
PC mode

DISPLAY, 4-11

Dynamic window, 1-3, 1-9
Recalculating DEFCENT automatically, 1-10

Developing Applications

Index

E

ECHO, 2-7
Dialogue Manager variable, 2-40
LET, 5-10

EDIT
Function
In Dialogue Manager, 2-30

EDUCFILE sample database, A-7
EMPDATA sample database, A-17
EMPLOYEE sample database, A-3
EMPTYREPORT parameter, 4-12

END command
Dialogue Manager, 2-39

ENDECHO, 5-10

Ending FOCUS sessions. See also FOCUS sessions

Entering FOCUS sessions. See also FOCUS sessions

Error messages, 6-12
Suppressing warning
FOC144, 4-17

ERRORS files

National language, 4-22

Euro support, 8-1
Calculations, 8-11
Currency conversion
Processing, 8-9
Currency database, 8-3
Currency-denominated field, 8-5

EUROFILE, 8-7

EVAL operator, 2-60

EXEC
Command, 2-38
From command line, 2-6, 2-86
Maximum nesting level, 2-5
Supplying values for variables, 2-67

Execution flow, 2-11, 2-15
-EXIT command, 2-12, 2-87
Index

EXPERSON sample database, A-18

Expressions
  Dialogue Manager conversion, 2-73
  -SET, 2-26, 2-33
  Computing new variables, 2-27
  Dates, 2-31
  DECODE, 2-29
  EDIT, 2-30
  Subroutines, 2-31, 2-33

External files
  Dialogue Manager example, 2-43
  DM example, 2-46
  NOCLOSE, 2-42, 2-69
  Reading, 2-69, 2-70
  Transferring to micros, 7-7
  Writing, 2-42, 2-106

External sort, 4-14

Extract files
  Listing current, 6-9
  SET parameters, 4-18

EXTSORT, 4-14

EXTTERM, 4-14

F

FDEFCENT, 1-3, 1-12

Field names
  66-character, 4-15
  Qualified, 4-15

Field Names Window, 3-7
  Goto values, 3-23

Field-level sliding window, 1-3, 1-12, 1-14

FIELDNAME
  SET, 4-15

Fields
  Continental Decimal Notation, 4-9
  In FOCUS file, 4-33
  Indexing, 4-33
    In FOCUS files, 4-17, 4-20, 4-21, 4-22
  Names
    Qualifying character, 4-32

EXPERSON sample database, A-18

Expressions
  Dialogue Manager conversion, 2-73
  -SET, 2-26, 2-33
  Computing new variables, 2-27
  Dates, 2-31
  DECODE, 2-29
  EDIT, 2-30
  Subroutines, 2-31, 2-33

External files
  Dialogue Manager example, 2-43
  DM example, 2-46
  NOCLOSE, 2-42, 2-69
  Reading, 2-69, 2-70
  Transferring to micros, 7-7
  Writing, 2-42, 2-106

External sort, 4-14

Extract files
  Listing current, 6-9
  SET parameters, 4-18

EXTSORT, 4-14

EXTTERM, 4-14

F

FDEFCENT, 1-3, 1-12

Field names
  66-character, 4-15
  Qualified, 4-15

Field Names Window, 3-7
  Goto values, 3-23

Field-level sliding window, 1-3, 1-12, 1-14

FIELDNAME
  SET, 4-15

Fields
  Continental Decimal Notation, 4-9
  In FOCUS file, 4-33
  Indexing, 4-33
    In FOCUS files, 4-17, 4-20, 4-21, 4-22
  Names
    Qualifying character, 4-32

File Contents Window, 3-4, 3-7
  Goto values, 3-23

File Directory Table, 6-5

File Names Window, 3-6
  Goto values, 3-23

FILEDEF
  ? FILEDEF, 2-23

File-level sliding window, 1-3, 1-12

FILENAME, 4-15

Files
  Extract
    Listing current, 6-9
    SET parameters, 4-18
    Listing number of segment instances, 6-7
    Listing time of last change, 6-7
    Percent of disorganization, 6-8
    Relating. See JOIN command
    Transferring, 7-9

FINANCE sample database, A-14

FIXFORM
  With sliding window, 1-5

FOC144
  Suppress warning, 4-17

FOCCOMP files, 7-4
  With sliding window, 1-6

FOCEXECs
  Loading into memory, 7-2

FOCSTACK, 2-4
  Limits, 4-17
  Processing procedures, 2-3, 2-5

FOCUS Database Server, 2-23, 7-12

FOCUS databases
  Physical characteristics, 6-5
  Transferring
    See XFER, 7-8
FOCUS utilities
   LET, 5-3
   LOAD, 7-2
   Query commands, 6-1
   Transferring files, 7-7

Format
   Specifications
      Variables, 2-73

Full-screen data entry, 2-46, 2-47
   Borders, 2-53, 2-111, 4-34
   -CRTFORM, 2-72

Functions and subroutines
   With sliding window, 1-5, 1-21, 1-28

FYRTHRESH, 1-3, 1-12

G

Global variable values, 2-50
   &&, 2-46

Global variables, 2-5
   && query command, 6-18

Global-level sliding window, 1-3, 1-6

-GOTO command, 2-16, 2-88

Goto values for windows
   Controlling transfers, 3-23
   Setting, 3-57
   Window applications, 3-3, 3-23, 3-57

Graphs
   Parameter settings
      ? SET GRAPH, 6-15
      ? SET GRAPH, 6-16

H

HiperCache, 4-7

HiperFOCUS
   Monitoring
      &HIPERFOCUS, 2-53, 2-112

HLIPRINT file, 6-8

HOLD files
   Controlling fields in, 4-18
   Listing fields in, 6-9
   Master File attributes for, 4-18

HOLDAATTR parameter, 4-18

HOLDLIST parameter, 4-18

HOLDSTAT parameter, 4-19

Horizontal menu, 3-5, 3-15
   Creating, 3-12
   Goto values, 3-14

Hot Screen facility
   Activating
      SET SCREEN, 4-34
   Help information
      Displaying, 4-19
      TOE, 6-19

HOTMENU, 4-19

I

-IF, 2-17
   Command, 2-90
   Loops, 2-27, 2-28

IF command
   AUTOSTRATEGY parameter, 4-5
   Key field search, 4-5

-IF…GOTO label
   Testing variables, 2-23

IMMEDTYPE, 4-21

Implied prompting, 2-47, 2-72, 2-73
   Format specifications, 2-73

IMS files
   SET, 4-21

-INCLUDE, 2-35, 2-91
   Nesting procedures, 2-5
   EXEC, 2-6

Index
   Fields
      Listing with ? FILE query command, 6-7
Index

INDEX, 4-4
   SET, 4-20, 4-22

Indexing fields
   Scheme, 4-20, 4-22

Information Builders system date, 1-30

Interactive procedures, 2-1

Internal matrix
   ? STAT query command, 6-16
   Preserving
      SAVEMATRIX, 4-33

Internal variables
   Testing, 2-23

J

JOBFILE sample database, A-6

JOIN command
   Listing active, 6-10

L

-label command, 2-92

LANG parameter, 4-22

LEFTMARGIN, 4-24

Legacy dates
   Validation, 1-6
   With sliding window, 1-4

LET, 5-1
   CLEAR, 5-11
      Global variables, 2-50
   Dialogue Manager and, 5-1
   ECHO, 5-10
   For non-English FOCUS users, 5-2
   In COMPUTE, 5-7
   In DEFINE, 5-7
   Listing substitutions, 6-11
   Placeholder symbols, 5-5
   SAVE, 5-12

LET (continued)
   Substitutions
      Multiple-line, 5-8
      Null, 5-7
      Recursive, 5-8
   Syntax
      Long form, 5-4
      Short form, 5-3
   Utilities, 5-1
   Variable substitutions, 5-6

Levels for sliding window, 1-3

Limits
   Dialogue Manager
      Nesting, 2-5, 2-35
      Storing variables internally, 2-73
   LET substitutions, 5-3

LINES parameter, 4-25

LOAD, 7-2
   ? LOAD, 7-5
   Loading
      Access Files, 7-3
      Compiled MODIFYs, 7-4, 7-5
      FOCEXECs, 7-3
      MFD, 7-3
      Removing files from memory, 7-5
      RUN, 7-4
      Types of LOAD files, 7-2
      UNLOAD, 7-5

Local variables, 2-5, 2-49

Long field names
   &FOCFIELDNAME, 2-52, 2-110
   &FOCQUALCHAR, 2-53, 2-111
   SET FIELDNAME, 4-15
   SET QUALCHAR, 4-32

Long form for LET, 5-4

Looping, 2-24, 2-28

LOTUS format
   Transfer to microcomputers, 7-7
Master File
- Database statistics, 6-7, 6-8
- Defining a sliding window, 1-12
- Loading into memory, 7-2
- Physical characteristics, 6-5
- Samples, A-2

Maximum
- Record length, 4-25
- MAXLRECL parameter, 4-25

Memory
- Cache, 4-7, 6-18

Menu
- Menu text, 3-14

Merge
- Routine invocations, 6-18
- MESSAGE parameter, 4-25

Messages
- Controlling display of, 4-25
- Displaying errors, 6-12
- Displaying in DM
  - -CRTFORM TYPE, 2-84
  - -TYPE, 2-10, 2-11, 2-103

Microcomputers
- Transferring files
  - Communication requirements, 7-7
  - File restrictions, 7-7
  - XFER, 7-9

Missing data
- ALL parameter, 4-2
- Controlling values for ACCEPT, 4-2
- Default symbol, 4-26
- NODATA character, 4-26

MODIFY
- Compiled requests, 7-1, 7-6
  - Loading into memory, 7-3
  - Loading into memory, 7-5
  - Defining a sliding window, 1-16
  - With sliding window, 1-5

Movies database, A-24

Developing Applications I-9

Index

MSG parameter, 4-25
Multi-Input window, 3-11
  - Creating, 3-16, 3-18
Multi-path files
  - Cartesian product, 4-8
Multiple-line LET substitutions, 5-8
-MVS RUN, 2-32, 2-93
MVS/TSO
  - PREFIX, 4-30
  - TRACKIO, 4-40

NA parameter, 4-26
Naming conventions
  - Reserved variables, 2-51, 2-54
National Language support
  - LANG query command, 6-11
  - Continental Decimal Notation, 4-9
  - DBCS
    - SHIFT parameter, 4-35
    - Status, 6-11
    - Support, 4-22
  - LET substitution, 5-2
  - Supported languages, 4-22

Nesting procedures
  - INCLUDE, 2-35, 2-91
  - Partial, 2-35, 2-91

NOCLOSE, 2-42, 2-69
NODATA parameter, 4-26
Non-FOCUS files. See also External files
  - In DM, 2-42, 2-69

NOPAGE, 4-27

Null
  - LET substitutions, 5-7

Numeric
  - Format
    - Dialogue Manager internal conversion, 2-73
Index

O

OFFLINE
  SET PRINT, 4-30
Offline printing
  SET, 4-30
OFFLINE-FMT, 4-26
ON
  TABLE, 4-1
ONLINE
  SET PRINT, 4-30
ONLINE-FMT, 4-27
Open-ended procedures, 2-39
Operating systems
  &FOCMODE, 2-52
Operators
  .EXIST, 2-20
  .LENGTH, 2-21
  .TYPE, 2-22
Optimization, 4-36
ORIENTATION, 4-27
Output Window, 6-19

P

Page layout parameters
  BOTTOMMARGIN, 4-7
  LEFTMARGIN, 4-24
  RIGHTMARGIN, 4-33
  TOPMARGIN, 4-39
  UNITS, 4-41
PAGE parameter, 4-27
PAGE-NUM parameter, 4-27
Pages
  Numbers, 4-27
PAGESIZE, 4-28
PANEL parameter, 4-28

Panels
  BYPANEL, 4-6
  COLUMNS, 4-9
PAPER parameter, 4-29
-PASS, 2-7, 2-94
PASS parameter, 4-29
PAUSE, 4-29
PAYHIST sample database, A-20
PF keys
  Setting, 4-30, 5-12
  Window Painter, 3-24
Placeholders, 5-5
Pointer chains, 6-6
Positional
  Variables, 2-66
Precedence for sliding windows, 1-3
PREFIX
  Setting for MVS data sets, 4-30
PRINT
  SET, 4-30
Procedures
  &ECHO, 2-40, 2-41
  &QUIT, 2-41, 2-59
Branching, 2-23
Calling
  EXEC, 2-38
Comment statements, 2-7
Concepts of execution, 2-5
Control statements, 2-8, 2-107
Controlling execution, 2-15
Creating, 2-6
Debugging, 2-40, 2-41
ECHO option, 2-7
Executing, 2-6
Exiting
  END, 2-39
  -QUIT, 2-15, 2-96
FOCSTACK diagram, 2-4
Incorporating multiple, 2-35
Locking and QUIT, 2-41
Developing Applications

Procedures (continued)
   Loop, 2-28
   Looping, 2-96
   Open-ended, 2-39
   Rules, 2-6, 2-108, 2-109
   Security, 2-7, 2-41
   Sending messages, 2-10, 2-11
   Setting return codes, 2-15, 2-23
   STACK, 2-40, 2-41
   Storing, 2-6
   TED, 2-6
   Variables in, 2-5, 2-46, 2-47

PROD, A-10

PROFILE
   Bypassing, 2-34

PROFILE FOCEXEC, 2-34

Prompting
   Dialogue Manager
      Direct with -PROMPT, 2-70, 2-95
   Implied, 2-47, 2-72
   Lists of value ranges, 2-74
   Testing supplied values, 2-23
   Variable values, 2-70

PTF query command, 6-13

Pulldown menus, 3-14
   Defaults, 3-14

Q

QUALCHAR parameter, 4-32

Qualified field names, 4-15
   DM, 2-53, 2-110, 2-111
   SET FIELDNAME, 4-15
   SET QUALCHAR parameter, 4-32
   SET QUALTITLES parameter, 4-32

QUALTITLES parameter, 4-32

Query commands, 6-1
   -? &, 2-48
   ? &&, 2-50, 6-18
   ? DEFINE, 6-3
   ? FDT, 6-5
   ? FILE, 6-7

Query commands (continued)
   ? FILEDEF, 2-23
   ? HOLD, 6-9
   ? JOIN, 6-10
   ? LANG, 6-11
   ? LET, 6-11
   ? n, 6-12
   -? parameter settings, 2-48
   ? SET, 6-14
   ? SET ALL, 6-14
   ? SET GRAPH, 6-15
   ? STAT, 2-58, 6-16
   ? SU, 2-23
   Status test, 2-23

-QUIT, 2-15, 2-96
   &QUIT, 2-41
   &QUIT FOCUS, 2-41, 2-96

R

-READ, 2-43, 2-69, 2-97

Reading
   Annotated DM example, 2-43, 2-46
   External files, 2-69, 2-70, 2-97
   -READ, 2-65

RECAP
   RECAP-COUNT parameter, 4-33
   RPAGESET parameter, 4-33

RECAP-COUNT parameter, 4-33

RECORDLIMIT parameter, 4-33

Records
   Maximum length, 4-25

Recursive
   LET substitutions, 5-8

REGION sample database, A-15

Remediation for cross-century dates, 1-1

REORG PERCENT, 6-8

-REPEAT command, 2-24, 2-98

Reporting overview
   Preview report, 4-41
Index

Reports
   Empty, 4-9
   ON syntax, 4-1
   Optimizing
      SQL Translator, 4-36

Reserved variables, 2-46, 2-51, 2-54, 2-59. See also Variables

Retrieval
   Indexed, 4-4
   Non-Indexed, 4-4
   Optimizing, 4-36

Return codes
   Setting in Dialogue Manager, 2-15
   Testing, 2-23

Return points
   Goto values, 3-23
   Return Value Display Window, 3-8
   Return values, 3-3, 3-8, 3-22

RIGHTMARGIN, 4-33

RPAGESET parameter, 4-33

RUN
   MODIFY, 7-4, 7-6

-RUN command, 2-11, 2-100

Run-time values
   Supplying, 2-46, 2-50
   Types of, 2-46

S

SALES sample database, A-8

Sample databases
   CAR, A-11
   COMASTER, A-21
   COURSES, A-16
   EDUCFILE, A-7
   EMPDATA, A-17
   EMPLOYEE, A-3
   EXPERSON, A-18
   FINANCE, A-14
   Movies, A-24
   PAYHIST, A-20
   PROD, A-10

Sample databases (continued)
   REGION, A-15
   SALES, A-8
   TRAINING, A-19
   VideoTrk, A-24

SAVE LET, 5-12

SAVEMATRIX, 4-33

SBORDER, 4-34

SCREEN, 4-34

Search string
   Window Painter, 3-1

Security
   SET PASS parameter, 4-29

Sequential data file
   Reading from, 2-43, 2-69, 2-97
   Writing to, 2-42, 2-43, 2-106

-SET, 2-26, 2-33, 2-65, 2-68, 2-101
   Calling subroutines, 2-31, 2-33
   Computing new variables, 2-27
   DECODE, 2-29
   EDIT, 2-30
   Loop, 2-28
   Query parameter, 2-48
   Supplying
      Lists of values, 2-74

SET parameters, 4-1
   ACCBLN, 4-2
   AGGRRATIO, 4-2
   ALL, 4-2
   ALLOWCVRERR, 1-29, 4-3
   ASNAMES, 4-3
   AUTOINDEX, 4-4
   AUTOPATH, 4-4
   AUTOSTRATEGY, 4-5
   AUTOTABLEF, 4-5
   BINS, 4-5
   BLKCALC, 4-5
   BOTTOMMARGIN, 4-7
   BUSDAYS, 4-6
   BYPANEL, 4-6
   BYSCROLL, 4-6
   CACHE, 4-7
   CARTESIAN, 4-8
Index

SET parameters (continued)
CDN, 4-9
COLUMNSCROLL, 4-9
COMPUTE, 4-10
COUNTWIDTH, 4-9
DATEDISPLAY, 1-29, 4-10
  With sliding window, 1-8
DATFNS, 4-10
DATETIME, 4-11
DEFCENT, 1-3, 1-6, 4-11
DISPLAY, 4-11
Displaying, 6-14
EMPTYREPORT, 4-12
ESTRECORDS, 4-12
EUROFILE, 4-13, 8-7
EXITAGGR, 4-13
EXITHOLD, 4-13
EXTSORT, 4-14
EXTTERM, 4-14
FIELDNAME, 4-15
FILE, 4-15
FILTER, 4-16
FIXRETRIEVE, 4-16
FOC144, 4-17
FOCALLOC, 4-17
FOCSTACK, 4-17
GRAPH
  ? SET GRAPH, 6-15
HDAY, 4-17
HLISUDUMP, 4-17
HLISUTRACE, 4-17
HOLDATTR, 4-18
HOLDLIST, 4-18
HOLDSTAT, 4-19
HOTMENU, 4-19
IBMLE, 4-20
IMMEDTYPE, 4-21
IMS, 4-21
INDEX, 4-20, 4-22
JOINOPT, 4-22
LANG, 4-22
LEADZERO, 4-24
LEFTMARGIN, 4-24
LINES, 4-25
MASTER, 4-25
MAXRECL, 4-25
MESSAGE, 4-25

SET parameters (continued)
MINIO, 4-26
NODATA, 4-26
OFFLINE-FMT, 4-26
ON TABLE, 4-1
ONLINE-FMT, 4-27
ORIENTATION, 4-27
PAGE-NUM, 4-27
PAGESIZE, 4-28
PAPER, 4-29
PASS, 4-29
PAUSE, 4-29
PFnn, 4-30
PREFIX, 4-30
PRINT, 4-30
PRINTPLUS, 4-31
QUALCHAR, 4-32
QUALTITLES, 4-32
Querying variable values, 2-48
REBUILDMSG, 4-32
RECAP-COUNT, 4-33
RECORDLIMIT, 4-33
RIGHTMARGIN, 4-33
RPAGESET, 4-33
SAVEMATRIX, 4-33
SBORDER, 4-34
SCREEN, 4-34
SHADOW, 4-35
SHIFT, 4-35
SORTLIB, 4-35
SPACES, 4-36
SQLOPTTTF, 4-36
SQUEEZE, 4-36
STYLESHEET (STYLE), 4-37
SUMPREFIX, 4-37
SUSI, 4-37
SUTABSIZE, 4-38
TEMPLDISK, 4-38
TERM, 4-22, 4-38
TESTDATE, 1-9, 1-30, 4-38
TEXTFIELD, 4-38, 4-39
TITLE, 4-39
TOPMARGIN, 4-39
TRACKIO, 4-40
TRMOUT, 4-40
TRMSD, 4-40
Index

SET parameters (continued)
  TRMSW, 4-40
  TRMTYP, 4-40
  UNITS, 4-41
  USER, 4-41
  WIDTH, 4-41
  XRETRIEVAL, 4-41
  YRTHRESH, 1-3, 1-6, 4-42

Setting
  Return values, 3-58

Shadow, 4-35

SHIFT parameter, 4-35

Short
  Form for LET, 5-3

Sink machine. See FOCUS Database Server

Sliding window. See also Dynamic window
  Implementing, 1-2
  Levels, 1-3
  MODIFY restrictions, 1-5
  Precedence, 1-3
  When to apply, 1-5

Smart dates. See Date formats

Sort
  External, 4-14, 6-17, 6-18
  Status of, 6-17

SPACES, 4-36

SQL date, 1-5

SQLTOPTTF parameter, 4-36

SQUEEZE parameter, 4-36

Stacked commands
  Executing with -EXIT, 2-12
  Executing with -RUN, 2-11

Statistical variables, 2-5, 2-19, 2-46, 2-57, 2-58
  ? STAT query, 2-58

STYLESHEET, 4-37

StyleSheets
  OFFLINE-FMT, 4-26
  ONLINE-FMT, 4-27
  ORIENTATION, 4-27
  PAGESIZE, 4-28
  STYLESHEET, 4-37

Subroutines
  Calling, 2-31, 2-33
  In DM, 2-31, 2-33
  Library (FUSELIB), 2-31

Substitutions
  LET
    Long form, 5-4
    Multiple-line, 5-8
    Null, 5-7
    Recursive, 5-8
    Short form, 5-3
    Variable, 5-5

Supplying values for variables, 2-47
  Amper variable values, 2-47
  -DEFAULTS, 2-85
  EXEC line, 2-67
  Expressions, 2-26, 2-33
  External files, 2-43, 2-69, 2-70
  Full-Screen (-CRTFORM), 2-47, 2-72, 2-84
  Implied prompting, 2-72, 2-73
  Limits, 2-108, 2-109
  -PROMPT, 2-70, 2-71, 2-95
  -READ, 2-69, 2-70, 2-97
  -SET, 2-26, 2-33
  Subroutines, 2-31, 2-33
  System variable values, 2-51, 2-54
  Verifying input, 2-73
  -WINDOW, 2-72, 2-104
  Without prompting, 2-65, 2-70, 2-72, 2-73

Suppressing
  Terminal output, 4-40

SUSI, 4-37

SUTABSIZE, 4-38

System date, 1-30

System variables, 2-5, 2-19, 2-51, 2-54. See also
  Variables
Index

T

TABLEF
  SQL Translator and, 4-36
TED
  Creating
    Procedures, 2-6
    TEXTFIELD, 4-39
TEMPDISK (TEMP), 4-38
Temporary fields
  With sliding window, 1-3, 1-24
TERM, 4-22, 4-38
Terminal
  Attributes, 4-14
  Specifying type, 4-38
  Support for screen borders, 4-34
  Suppressing output, 4-40
Terminal Operator Environment (TOE)
  Displaying
    Output, 6-19
  Hot Screen facility, 6-19
  Line mode output
    IMMEDTYPE, 4-21
  Windows
    Output, 6-19
Testing
  Logic, 2-41
  Status of a query, 2-23
  Variables
    Input value, 2-73
    List of ranges, 2-74
Text Display Window, 3-6
  Goto values, 3-23
Text Input Window, 3-5
  Goto values, 3-23
Text line (menu feature), 3-14
TEXTFIELD, 4-38, 4-39
Threshold. See YRTHRESH
Time stamp, 1-30
TITLE parameter, 4-39
TOPMARGIN, 4-39
TRACKIO, 4-40
TRAINING, A-19
Transactions
  Displaying last change, 6-7
TRMOUT, 4-40
TRMSW, 4-40
TRMTYP, 4-40
-TSO RUN, 2-102
  Subroutines, 2-33
Tutorial
  Window Painter, 3-27, 3-47
TYPE
  -CRTFORM, 2-84
  -TYPE, 2-10, 2-11, 2-103
U

UNITS parameter, 4-41
UNLOAD, 7-5
USER, 4-41
Utilities
  JOIN
    ? JOIN query command, 6-10
  LET, 5-3
    ? LET query command, 5-9, 6-11
    LET CLEAR, 5-11
    LET SAVE, 5-12
  Query commands, 6-1
  SET parameters, 4-1
    ? SET query command, 6-14
Variables (continued)

Verifying
- Format specifications, 2-73
- Lists of ranges, 2-74
- Status of a query, 2-23
- Variable values, 2-73

Vertical Menu, 3-4
- Goto values, 3-23

VideoTrk database, A-24

Virtual fields
- With sliding window, 1-3, 1-19, 1-22

W

WHERE
- AUTOSTRATEGY parameter, 4-5

Width
- Maximum for report, 4-40

WIDTH parameter, 4-41

-WINDOW, 2-47, 2-72
- Syntax, 2-104, 3-19

Window applications. See Window Painter and Terminal Operator Environment

WINDOW COMPILE, 3-70, 3-77, 3-78

Window files, 3-2

Window Options Menu, 3-56
- Description, 3-56
- Display list, 3-59
- Goto value, 3-57
- Heading, 3-59
- Help window, 3-62, 3-66
- Hide list, 3-60
- Line break, 3-64
- Multi-select, 3-65
- Popup, 3-61
- Quit, 3-66
- Return value, 3-58
- Show a window, 3-59
- Unshow a window, 3-59

WINDOW PAINT, 3-48
Index

Window Painter, 3-1
  Activating, 3-2
  Creating
    Horizontal menu, 3-12, 3-15
    Multi-Input window, 3-16, 3-18
  DM, 2-105, 2-109, 3-19
  Entry Menu, 3-31
  Invoking, 3-48
  Main Menu, 3-10, 3-33, 3-50
  Menu text, 3-14
  Screens, 3-26, 3-27, 3-47
  String search, 3-1
  Testing PFkey values, 3-24
  Text line, 3-14
  Transferring files, 3-69
  Tutorial, 3-27, 3-47
  Utilities Menu, 3-67
    Change file description, 3-69
    Compress the file, 3-69
    Copy a window, 3-69
    Create a transfer file, 3-69
    Document the file, 3-68
    Rename a window, 3-69
  WINDOW COMPILE, 3-70, 3-77, 3-78
  Window Creation Menu, 3-52
  Window Design Screen, 3-54
  Window files, 3-70
    CMS, 3-70
    MVS/TSO, 3-70
  WINDOW PAINT, 3-2
  Window transfer files, 3-69
  Windows
    Creation Menu, 3-52
    Design Screen, 3-53
    Entry Menu, 3-31
    Execution, 3-9
    Field Names, 3-7
    File Contents, 3-7
    File Names, 3-6
    Horizontal menu, 3-5

Window Painter

Windows (continued)
  Main Menu, 3-10, 3-50
  Multi-Input window, 3-11
  Pulldown menus, 3-14
  Return Value Display, 3-8, 3-9
  Text Display, 3-6
  Text Input, 3-5
  Vertical Menu, 3-4

-WINDOW statement, 3-19
-WRITE, 2-42, 2-43, 2-106

X

XFER
  Downloading files, 7-8
  Uploading files, 7-9

XRETRIEVAL, 4-41

Y

Y2K. See Sliding window

Year 2000
  Displaying a date variable without separators, 2-56
  YRTHRESH
    As offset from current year, 4-42

Year threshold. See YRTHRESH

YRTHRESH, 1-2, 1-6
  As offset from current year, 4-42
  In Master Files, 1-12
  Negative values, 1-3
  Querying
    Current global value, 1-11
    Value in Master Files, 1-30
  With COMPUTE, 1-24, 1-26
  With DEFINE, 1-3, 1-19
Reader Comments

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

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. Send comments to:

Corporate Publications
Attn: Manager of Documentation Services
Information Builders
Two Penn Plaza
New York, NY 10121-2898

or FAX this page to (212) 967-0460, or call Sara Elam at (212) 736-4433 x.3207

Name: _____________________________________________________________________________________
Company: __________________________________________________________________________________
Address: ___________________________________________________________________________________
Telephone: _____________________________________________ Date: ______________________________
Comments:

Name: _____________________________________________ Date: ______________________________