ibi™ FOCUS®

Studio Installation and User Guide

Version 9.3.0 | April 2024

Contents

Installing ibi FOCUS with FOCUS Studio

ibi™ FOCUS® and FOCUS Studio are installed by going to the eDelivery site, downloading the specific .run file to a USS directory, and then using the sh command to execute the .run file.

By selecting your product, version, and operating system, and accepting the EULA agreement, you may select to download the full product or individual files. If you choose individual files, you must then open the FOCUS® Software folder, select an IBI_focus_*_mvs_zseries.run file, where * indicates the release number, and start the download. The IBI_focus_*_mvs_zseries.run file is a self-extracting .run archive format file.

Once the desired .run file is downloaded and transferred to the actual machine where the installation will occur and into a temporary USS working directory, execute the .run file.

More specifically, change directory (cd) to the temporary directory and issue the following command, where the actual .run file name is the full file name that was downloaded.

```
sh IBI_focus_*_mvs_zseries.run
```

During the execution of the .run file, you will be required to supply a high-level-qualifier (HLQ) string using a command line parameter or in response to a prompt. The default HLQ is the running user-id. Execution of the .run file will test whether existing libraries will be overwritten, and if so, it will prompt whether to proceed or quit. Successful execution of the .run file will create eight PDSE libraries starting with *hlq*.F.HOME.

The extracted files will be included in a temporary subdirectory of the current directory. After successful execution of the .run file, you need to run ISETUP from the *hlq*.HOME.DATA library in zOS. You must use the ISETUP procedure in order to install or apply maintenance to FOCUS.

Install FOCUS with FOCUS Studio

Procedure

1. Change directory (cd) to the temporary USS directory where the installation will occur.
2. Issue the following command, where the actual .run file name is the full file name that was downloaded. For example:

   ```
   sh IBI_focus_9.3.0_mvs_zseries.run
   ```

   The following prompt appears:

   ```
   Enter datasets HLQ prefix:
   ```

3. Type your HLQ, for example:

   ```
   PMSSAE.MVS1.DF9999
   ```

   - If the datasets already exist, you will receive the following messages, along with a prompt asking whether you want to proceed:

     ```
     Installation datasets prefixed with PMSSAE.MVS1.DF9999 already
     exist.
     They will be deleted and re-created.
     Do you want to proceed (Y/N)?
     ```

     If you type *Y* to preceed, you will receive the following messages that all the temporary datasets are allocated and restored successfully:

     ```
     ALL TEMPORARY DATASETS ARE ALLOCATED SUCCESSFULLY ->
     PROCESSING
     ALL DATASETS ARE RESTORED SUCCESSFULLY
     ```

   - If the datasets do not already exist, the datasets will be allocated and you will receive a message that the datasets were allocated successfully.

Result

Sample Output With Responses

```
sh IBI_focus_9.3.0_mvs_zseries.run
Enter datasets HLQ prefix: PMSSAE.MVS1.DF9999
Installation datasets prefixed with PMSSAE.MVS1.DF9999 already exist.
They will be deleted and re-created.
Do you want to proceed (Y/N)? Y
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.BIN NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.ACX NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.FEX NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.MAS NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.ERR NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.LOAD
NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.DATA
NOT IN CATALOG OR CATALOG CAN NOT BE ACCESSED
13.33.11 STC47294  +IKJ56228I DATA SET PMSSAE.MVS1.DF9999.P.HOME.ETC NOT
IN CATALOG OR CATALOG CAN NOT BE ACCESSED
ALL TEMPORARY DATASETS ARE ALLOCATED SUCCESSFULLY -> PROCESSING
ALL DATASETS ARE RESTORED SUCCESSFULLY
```

After extraction, proceed with the instructions in Introduction to ISETUP.

Introduction to ISETUP

The ISETUP procedure simplifies the installation process by:

- Requiring less interaction.
- Automating most of the procedure.
- Providing a log of all the activity that takes place, to allow for verification.

ISETUP asks you for the high-level qualifier for your FOCUS data sets. Based on this qualifier and a list of all of the standard data set names (low level qualifiers), ISETUP installs all the necessary FOCUS libraries. The list of low level qualifiers ISETUP uses is in member FOCSNAME in the F.HOME.DATA data set. Although it is not advisable, it is possible

to modify the low-level qualifiers of the files. For more information, see Specifying Non-Standard Data Set Names.

About the ISETUP Procedure

The ISETUP procedure is an interactive process that uses ISPF panels.

Important! You must use the version of ISETUP that comes with your FOCUS release to install FOCUS.

Installing without the use of ISETUP is no longer supported. Additionally, you cannot use previous versions of your installation JCL to install FOCUS.

You must execute ISETUP from the ISPF command shell.

You should not use ISETUP to overwrite your current production FOCUS data sets with new versions.

Note: After running ISETUP, you will have to install the new FOCUS License key in order to verify your installation. For more information on requesting and installing your FOCUS license key, see ibi FOCUS License Keys.

Overview of the ISETUP Procedure

This topic provides a high-level overview of the installation procedure and the steps that may be needed after running the installation procedure.

1. Execute the ISETUP procedure from the ISPF command shell.
    a. Fill in the information requested on the screens displayed by ISETUP.
    b. Submit the JCL created by ISETUP when ready to proceed.
    c. Verify that the ISETUP job was successful by examining the completion codes for each step.

    When this is complete, you have a basic test environment with the FOCUS installation or maintenance applied which is ready to be tested using the verification standards of your site.
2. After successful testing of the FOCUS release or maintenance, move the test

environment to production according to the standards of your site.

Overview of Manual Steps Following ISETUP

The following features are not installed by ISETUP. For these features, you must follow the instructions in the FOCUS Installation Guide for your current production version of FOCUS:

- The number of cache pages.
- FOCPARM member of ERRNLS.DATA data set.
- FOCPROF member of ERRNLS.DATA data set.
- IBITABLA member of FOCCTL.DATA data set.
- The FOCUS Menu.
- The FOCUS Toolkit.
- Simultaneous Usage facility (SU).
- User Exits.
- IBI Subsystem (the Subsystem name, however, is preserved).
- HiperBudget.
- CA-ACF2 Interface.
- SU Security Interface (SUSI).
- NLS configuration.
- Maximum number of data exceptions.
- Data Adapters (Interfaces, for example, Db2 or ADABAS).

  Important: All data adapter files are included and allocated in FOCUS once you run ISETUP. However, using the instructions in the relevant Installation Guides for those data adapters, you must reinstall every data adapter to which you want to have access in the installed version of FOCUS.

If you run FOCUS out of LPA libraries, you must do the following:

1. Copy all the reentrant modules back into FOCLIB.LOAD from FOCLPA.LOAD. (Use JOB JFSCPBCK.)

2. Run ISETUP.
3. Copy the reentrant modules from FOCLIB.LOAD back into FOCLPA.LOAD. (Repeat JOB JFSCPLPA.)
4. Delete the reentrant modules FROM FOCLIB.LOAD. (Repeat JOB JFSDELPA.)

Installing ibi FOCUS Using ISETUP

The ISETUP panels consist of **input** fields in which you can enter information and **display** fields in which you cannot enter information. Input fields have a red background and display fields have a white background.

Note: The ISETUP procedure is in a library named HOME.DATA. You should receive all data sets to a name that keeps the existing low-level qualifiers and prefixes them with a high-level qualifier of your choice, followed by a required FOCUS qualifier consisting of the letter F. For example, HOME.DATA will become **new_hql**.F.HOME.DATA.

After the ISETUP procedure has executed, all of your HOME libraries will have names of the form **new_hlq**.F.HOME.**suffix**, where **new_hlq** is the high-level qualifier you specify on the ISETUP panels. For example, if you specify the high-level qualifier FOCUS.*nnnn*, ISETUP will create a data set called FOCUS.*nnnn*.F.HOME.LOAD.

The installation procedure builds and, optionally, executes the JCL needed to install FOCUS.

Note that when a screen opens, its input fields may be populated with the values from a previous installation. When you change any values, you must press **Enter** once to register the changes and then press **Enter** again to proceed to the next panel.

If you decide you are not satisfied with the values you entered and you want to change them, press **PF3** to return to the previous panel.

ISETUP Processing Flow

After you download and receive the HOME.DATA library, you have the ISETUP installation utility on your system. When you execute ISETUP, the installation job should complete with 0 return codes for all steps, and 39 additional libraries will have been created under the high-level qualifier you entered on the ISETUP screen (for a total of 40 libraries).

There will be 26 logical alias libraries:

```
hlq.ADABAS.DATA                                 *ALIAS
hlq.ADABAS.LOAD                                 *ALIAS
hlq.DATACOM.DATA                                *ALIAS
hlq.DATACOM.LOAD                                *ALIAS
hlq.ERRNLS.DATA                                 *ALIAS
hlq.ERRORS.DATA                                 *ALIAS
hlq.FOCCTL.DATA                                 *ALIAS
hlq.FOCDBC.DATA                                 *ALIAS
hlq.FOCDBC.LOAD                                 *ALIAS
hlq.FOCEXEC.DATA                                *ALIAS
hlq.FOCLIB.LOAD                                 *ALIAS
hlq.FOCM204.DATA                                *ALIAS
hlq.FOCM204.LOAD                                *ALIAS
hlq.FOCSTYLE.DATA                               *ALIAS
hlq.FOCSQL.DATA                                 *ALIAS
hlq.FOCSQL.LOAD                                 *ALIAS
hlq.FUSELIB.DATA                                *ALIAS
hlq.FUSELIB.LOAD                                *ALIAS
hlq.IDMS.DATA                                   *ALIAS
hlq.IDMS.LOAD                                   *ALIAS
hlq.IMS.DATA                                    *ALIAS
hlq.IMS.LOAD                                    *ALIAS
hlq.MASTER.DATA                                 *ALIAS
hlq.MODEL.DATA                                  *ALIAS
hlq.TRF.DATA                                    *ALIAS
hlq.WINFORMS.DATA                               *ALIAS
```

There will be eight physical HOME libraries

```
hlq.F.HOME.BIN                                  USERM1
hlq.F.HOME.FEX                                  USERM1
hlq.F.HOME.ACX                                  USERM1
hlq.F.HOME.LOAD                                 USERM1
hlq.F.HOME.ERR                                  USERM1
hlq.F.HOME.ETC                                  USERM1
hlq.F.HOME.DATA                                 USERM1
hlq.F.HOME.MAS                                  USERM1
```

There will be six physical CONF libraries:

```
hlq.CONF.ACX                                    USERM1
hlq.CONF.CFG                                    USERM1
hlq.CONF.DATA                                   USERM1
hlq.CONF.SQL                                    USERM1
hlq.CONF.MAS                                    USERM1
hlq.CONF.PRF                                    USERM1
```

To run the newly installed release of FOCUS, you can use your CLIST or JCL from a prior release of FOCUS by changing the high-level qualifier to the one you just installed. You *do not* have to modify the CLIST or JCL to allocate any CONF or HOME libraries. This is the reason the alias libraries were created.

Note: There is a sample CLIST (member FOCUSC), JCL (member FOCUS), and alias creation job (member ALIAS) in the CONF.DATA library.

Installing ibi FOCUS Using the Disk Option

This section describes how to install FOCUS using distribution files already on your system (Option D).

Option D enables you to create a new environment without having to download the FOCUS files again. For example, you may want to replicate a test environment under another high-level qualifier or replicate a production environment to move to another location.

Optionally, Specify Non-Standard Data Set Names

ISETUP asks you for the high-level qualifier for your FOCUS data sets. Based on this qualifier and a list of all of the standard data set names (low level qualifiers), ISETUP installs all the necessary FOCUS libraries. The list of low level qualifiers ISETUP uses is in member FOCSNAME in the F.HOME.DATA data set. If you do not want to use the standard data set names, see Specifying Non-Standard Data Set Names.

Invoke ISETUP to Install ibi FOCUS With the Disk Option

Now you can invoke the ISETUP procedure to create a new instance of FOCUS using your previously installed version of FOCUS.

Invoke ISETUP to Install ibi FOCUS With the Disk Option

Invoke ISETUP by executing the CLIST contained in member ISETUP of the *installed_hlq*.F.HOME.DATA data set, where *installed_hlq* is the high-level qualifier for the previously unloaded or downloaded version of HOME.DATA.

Note: You must use the version of ISETUP that you downloaded or unloaded with the FOCUS data sets from which you are creating the new instance. You cannot use an older version of ISETUP.

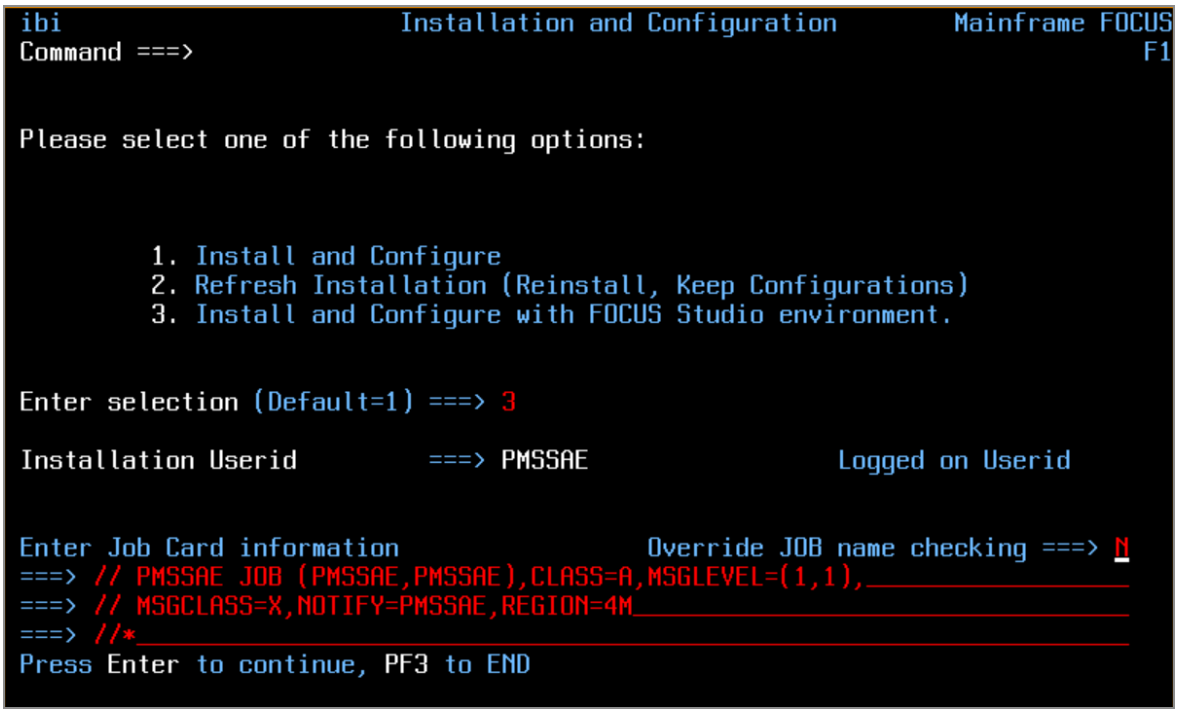
Procedure

1. You must execute this CLIST from the ISPF command shell (option 6):

   ```
   EXEC 'installed_hlq.F.HOME.DATA(ISETUP)'
   ```

   The FOCUS Installation and Configuration panel opens.

2. Choose Option **3** to install and configure FOCUS with FOCUS Studio, as shown in the following image.



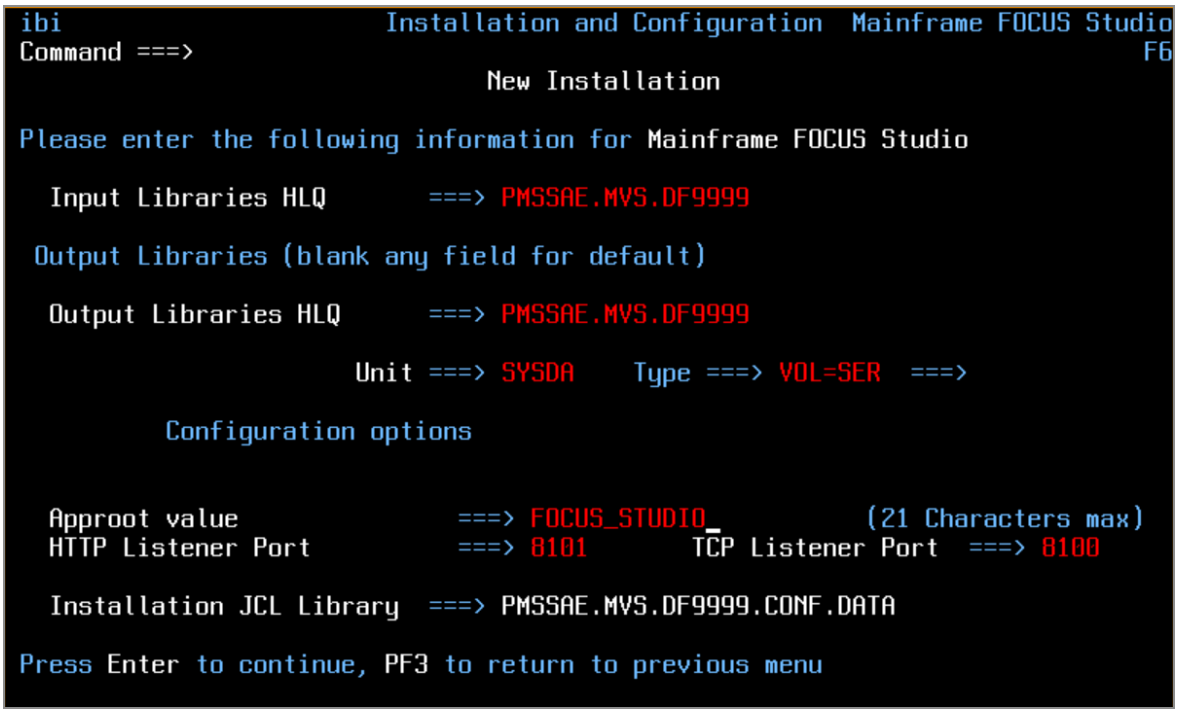
The job card information will be filled in with the options used in your previous installation, and the **Installation Userid** field will contain your logon ID.

Note: Your JOB card must have a REGION parameter that specifies at least 4M. If you get a message that the region size is not large enough to run ISETUP, change the REGION parameter to specify 0M and rerun ISETUP.

3. Press **Enter** to register this choice.
4. Press **Enter** again to proceed to the next panel.

   The New Installation FOCUS Library Definitions panel opens, as shown in the following image.

   The input and output libraries will be the values used in your previous installation.



5. Tab to the input fields and type the high-level qualifier for the output libraries that will be created by ISETUP.

   You can change the following input fields:

   - **Input Libraries HLQ.** Type the high-level qualifier of the installed FOCUS libraries.
   - **Output Libraries HLQ.** Type the high-level qualifier for the additional set of FOCUS libraries.

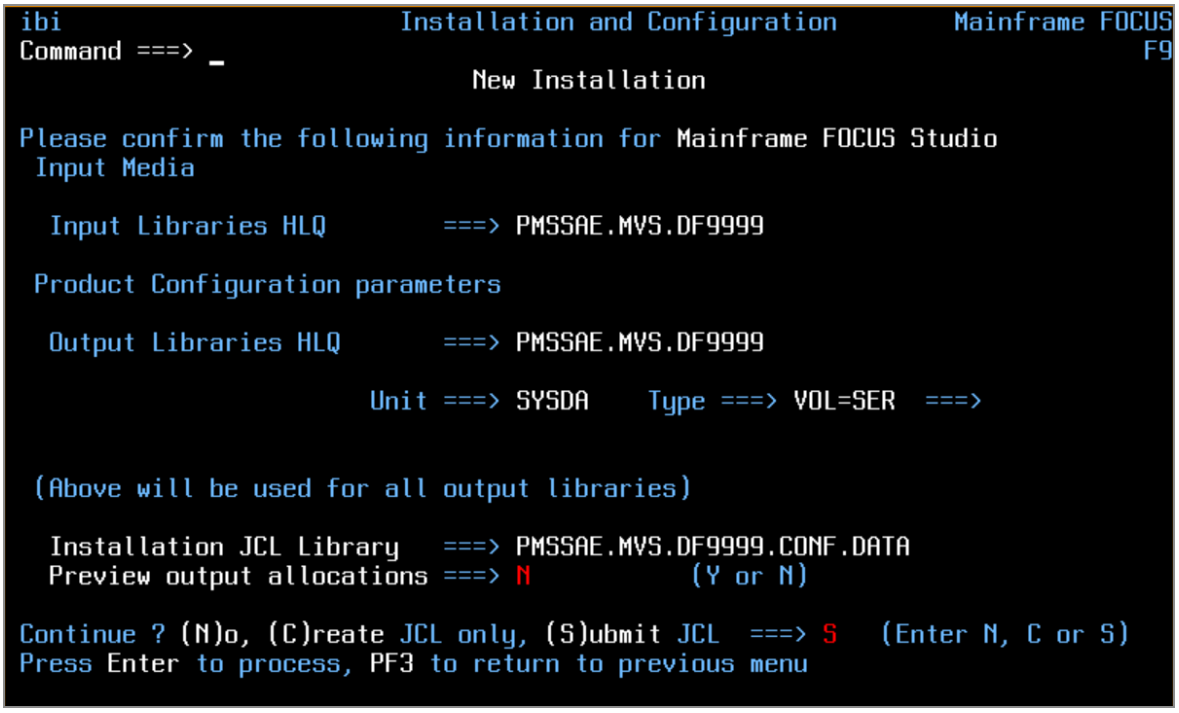
     Note: You must make sure that the combination of your high-level qualifier and the FOCUS library names is less than or equal to 44 characters, the maximum name length supported on z/OS.
   - **Unit.** By default, the **Unit** value is **SYSDA**. SYSDA is defined by your systems programming group to be a type of DASD device. FOCUS must be installed into PDSE libraries on a 3390 device type. If the SYSDA definition at your site does not comply with these requirements, change the **Unit** value to an appropriate DASD subpool name that defines a 3390 PDSE library.
   - **Volume.** This is an optional input value that can be used to place the new FOCUS libraries on a specific volume.

- **Approot value.** Type an Approot value for access to the product components.

When you are finished, press **Enter**.

Press **Enter** again to proceed to the next panel.

The New Installation FOCUS Library Confirmation panel opens, as shown in the following image.



6. Confirm the options that you entered on previous panels.

   You can see what the installation process will allocate on your behalf by changing the **Preview output allocations** field from **N** to **Y** and pressing **Enter**.

   If you changed **Preview output allocations** to **Y**, the preview panels will display before ISETUP creates the installation JCL. While you examine the Preview panels, if you see one or more values you want to correct, press **PF3** as many times as necessary to return to the panel that you want to modify.

   There may be several preview panels. You can navigate between them using the following keys:

   - To continue to the next preview panel, press **PF8**.
   - To return to the previous preview panel, press **PF7**.

- To exit the preview panels and return to the **Confirmation** panel, press **Enter** or **PF3**.

After you return from the preview panels to the **Confirmation** panel, the **Preview output allocations** field is reset to **N**.

7. When you have verified your choices on the Confirmation panel, you must choose a continuation option:
   - The default option is **N**, which cancels the process and returns to the FOCUS Library Installation panel.
   - Option **C** creates the installation JCL and saves it to member ISETUPJ1 in the NEW_HLQ.CONF.DATA data set. You can review, edit, update, or submit it manually without reinvoking ISETUP:

     ```
     ISETUP - Allocating file 'NEW_HLQ.CONF.DATA'
     ISETUP - Creating installation options file ISOPTS1
     ISETUP - Creating main JCL ISETUPJ1
     ISETUP - JCL created in NEW_HLQ.CONF.DATA but not submitted
     ISETUP - You can manually submit the following JCL to complete
     install
     ISETUP - NEW_HLQ.CONF.DATA(ISETUPJ1)
     ***
     ```

   - Option **S** creates the installation JCL, saves it to member ISETUPJ1 in the NEW_HLQ.CONF.DATA data set, and submits the job directly to the system for processing:

     ```
     ISETUP - Deleting file  'NEW_HLQ.CONF.DATA'
     IDC0550I ENTRY (A) NEW_HLQ.CONF.DATA DELETED
     ISETUP - Allocating file 'NEW_HLQ.CONF.DATA'
     ISETUP - Creating installation options file ISOPTS1
     ISETUP - Creating main JCL ISETUPJ1
     ISETUP - Submitting JCL, use SDSF (or site standard) to monitor
     job
     IKJ56250I JOB USERID1I(JOB28173) SUBMITTED
     ***
     ```

   To register your choice, press **Enter**.

   To execute the option you chose, press **Enter** again.

ISETUP Messages

A message similar to the following is issued when the high-level qualifier is too long (when concatenated to the name of the installation library):

```
IKJ56709I INVALID DATA SET NAME,
'QCSPDS.R729999B.EDAPORT.BF9999.DATACOM.DATA'
IKJ56709I INVALID DATA SET NAME,
'QCSPDS.R729999B.EDAPORT.BF9999.FOCM204.DATA
```

The following warning message displays when libraries are about to be replaced:

```
Warning - One or more product files
already exist. Press enter to continue
(files will be deleted) or PF3 to
return to previous panel to provide a
different output high level qualifier.
```

The following confirmation message displays when libraries will be deleted:

```
Confirmation - Are you sure you want to
continue? Files will be deleted.
Press enter to confirm or PF3 to return
to previous screen.
```

ibi FOCUS License Keys

As of January 1st, 2023, the FOCUS licensing process and keys for FOCUS licenses have changed.

As a result of these changes, you need a new FOCUS license key if you are:

- Renewing your contract.
- Upgrading your FOCUS release.
- Upgrading or changing your hardware.
- Changing your CPU serial number.
- Requesting a disaster recovery key.

To request a new license key, open a Customer Support case and supply the following information:

- Output from the "? CPUID" command issued from the FOCUS command line of your current version of FOCUS.
- Output from the "D M=CPU" system command.
- Output from the "? REL" command.
- Expiration date of your current or renewed contract.

Once we receive all the necessary information, we can provide a new license key.

If you are using FOCUS...	You will...
Release 7.7.03M through Release 9.1.1	Receive a standardized IBICPUID load module, which you will add to your FOCLIB.LOAD library. Note that a new site code may be provided. This module will be zapped with your new CPUPLATE. (Note that we will provide JCL to zap the new site code, if required.) You will also receive a PTF that must be run. The PTF contains JCL and object code to call the IBICPUID module.
Releases prior to Release 7.6.13	Receive a standardized IBICPUID load module, which you will use to replace your existing module. Note that a new site code may be provided. This module will be zapped with your new CPUPLATE. (Note that we will provide JCL to zap the new site code, if required.)

Note: Information on the new ibi™ FOCUS® License Management facility and installation instructions for installing the CPUPLATE are included in *New ibi™ FOCUS® License Management for z/OS Sites*.

New ibi FOCUS License Management for z/OS Sites

ibi FOCUS has a new license management facility, which is a mechanism that registers your copy of FOCUS.

This facility checks that your site:

- Has registered the FOCUS software.
- Does not exceed its current expiration date.
- Is running on the registered hardware.

If any of these is not true, or the registration has not been done, an appropriate warning or violation message is displayed each time a user enters FOCUS, and FOCUS will not start.

This new FOCUS License Management facility requires a new version of the IBICPUID load module that is a member of FOCLIB.LOAD. The first time that you request a new key, this new module will be zapped with your information and shipped to you. You will only need to replace the existing IBICPUID module or add the IBICPUID module to FOCLIB.LOAD.

If you need a new CPUPLATE for any reason, you will be required to submit the information indicated below and a new IBICPUID module will be created and sent to you.

New License Overview

The new license management facility is a processor, expiration date, and site code registration facility.

- **Processor registration.** The instructions for installing the CPUPLATE are contained in this document. ibi takes the required information and generates an encrypted registration ID called a CPUPLATE.
    - If your processor has not been registered, a warning message is displayed at FOCUS initialization and FOCUS will not start up.
    - If your copy of FOCUS is registered and then run on a processor other than the one on which FOCUS was registered, or has exceeded the registered expiration date, or if any error occurred during processor registration, a violation message is displayed and FOCUS will not start up.
    - You will need to contact ibi Customer Support and request a new CPUPLATE.

  In order to generate a CPU registration ID, ibi needs the following information:
    - Output from the "? CPUID" command issued from the FOCUS command line of your current release of FOCUS.

- Output from the "D M=CPU" system command.
- Output from the "? REL" command.
- Expiration date of your current or renewed contract.

- **Site code registration.** The instructions for installing the site code are documented in the *OS/390 and MVS Installation Guide*. We will be using your existing ibi site code, if possible. If required, we will send JCL to zap your site code.

Obtain the Required CPU information

Procedure

1. To obtain the CPU ID, issue the following command from the FOCUS command prompt:

   ```
   ? CPUID
   ```

   This command displays the following information.

   ```
   OBSERVED CPU:
      *** Processor mmmm Model nnnn-vv Max pp Site ssss.ss
          LICENSED CPU(S)
      *** NONE
   ```

   where:

   mmmm

   Is the Processor ID.

   nnnn

   Is the model number.

   vv

   Is the version number.

   pp

   Is the number of processors on the system.

ssss.ss

Is the site code.

2. You also need to capture the information from the operator console when issuing the following z/OS operator command:

```
D M=CPU
```

This command displays the following information.

```
IEE174I 12.16.10 DISPLAY M 218
```

```
PROCESSOR STATUS
```

```
ID   CPU          SERIAL
```

```
1     +           11115379121
```

```
2     +           21115379121
```

```
3     +           31115379121
```

```
4     +           41115379121
```

3. To run ? REL:
   - Start FOCUS.
   - From the FOCUS command prompt, issue the following command:

     ```
     ? REL
     ```

4. Issue the appropriate commands to obtain the required CPU information.
5. Open a Support Case with Customer Support.
6. Provide the CPU information for all CPUs to the Customer Support Consultant. They

will provide you with the CPUPLATE ID for each CPU.

Installation Instructions for Release 7.6.13 and Earlier

1. Download the zip file from the Support Case.
2. Expand the files in the zip file to a temporary directory.

   For Release 7.0.8R through Release 7.6.09, you will have the following files:

   - foclib.xmit
   - jcl.xmit

   For Release 7.6.10 through Release 7.6.13, you will have the following files:

   - foclib.xmit
   - jcl.xmit
   - flicense.data
3. On your mainframe, allocate files to receive the files from the zip, as follows:

   ```
   userid.IBIKEYS.LOAD.TEMP
   LRECL(80),RECFM(F,B),BLKSIZE(3120),SPACE=(TRK,(5,5))
   ```

   ```
   userid.IBIKEYS.JCL.TEMP
   LRECL(80),RECFM(F,B),BLKSIZE(3120),SPACE=(TRK,(5,5))
   ```

   Note: For FOCUS Release 7.6.10 through Release 7.6.13 only:

   ```
   userid.IBIKEYS.FLICENSE.DATA,DISP=(,CATLG,DELETE),LIKE
   focuslibrary.ERRORS.DATA
   ```
4. Using FTP, transfer the following files to the temporary files you just created, as follows:

   ```
   binary
   put jcl.xmit 'userid.IBIKEYS.JCL' (REPLACE
   ```

```
put foclib.xmit 'userid.IBIKEYS.LOAD.TEMP' (REPLACE
```

Note: For FOCUS Release 7.6.10 through Release 7.6.13 only:

```
ascii
put flicense.data 'userid.FLICENSE.DATA(FLICENSE)' (REPLACE
quit
```

5. Using TSO, expand the LOAD.TEMP dataset as follows:

   ```
   RECEIVE INDSN('userid.IBIKEYS.LOAD.TEMP')
   ```

   Reply to the restore prompt with

   ```
   DSNAME('userid.IBIKEYS.LOAD')
   ```

6. Using TSO, expand the JCL.TEMP dataset as follows:

   ```
   RECEIVE INDSN('userid.IBIKEYS.JCL.TEMP')
   ```

   Reply to the restore prompt with

   ```
   DSNAME('userid.JCL.DATA')
   ```

7. Make a backup of your FOCLIB.LOAD library.
8. Copy and replace the IBICPUID module from '*userid*.IBIKEYS.LOAD' to your FOCLIB.LOAD library.
9. Edit '*userid*.IBIKEYS.JCL' by adding a job card and updating the JCL SET to identify your FOCLIB.LOAD library.

   Note: This will update the site code.
10. Submit the JCL. This will apply the new site code.
11. If you are running Release 7.6.10 through Release 7.6.13, replace your current FLICENSE member with the one from the zip file.
12. If you have FOCUS loaded in LPA and/or linklist, you will need to refresh them with this newly zapped module. This is a systems task and should be referred to your systems group.

13. Verify your FOCUS license key installation by running FOCUS and issuing the ? CPUID command in both interactive and batch environments. If FOCUS starts then your installation has been successful. If FOCUS does not start, then the installation has not been successful. Please open a case with Customer Support to review the parameters that have been provided and the process that has been followed.

Installation Instructions for Release 7.7.03 and Higher

1. Download the zip file from the Support Case.
2. Expand the files in the zip file to a temporary directory.

   For Release 7.7.03 and higher, you will have the following files:

   ```
   foclib.xmit
   ```

   ```
   jcl.xmit
   ```

   ```
   flicense.data
   ```

3. On your mainframe, allocate files to receive the files from the zip, as follows:

   ```
   userid.IBIKEYS.LOAD.TEMP
   LRECL(80),RECFM(F,B),BLKSIZE(3120),SPACE=(TRK,(5,5))
   ```

   ```
   userid.IBIKEYS.JCL.TEMP
   LRECL(80),RECFM(F,B),BLKSIZE(3120),SPACE=(TRK,(5,5))
   ```

   ```
   userid.IBIKEYS.FLICENSE.DATA,DISP=(,CATLG,DELETE),LIKE
   focuslibrary.ERRORS.DATA
   ```

4. Using FTP, transfer the following files to the temporary files you just created, as follows:

```
ascii
put flicense.data 'userid.IBIKEYS.FLICENSE.DATA(FLICENSE)'
binary
put foclib.xmit 'userid.IBIKEYS.LOAD.TEMP' (REPLACE
put jcl.xmit 'userid.IBIKEYS.JCL.TEMP' (REPLACE
quit
```

5. Using TSO, expand the LOAD.TEMP dataset as follows:

   ```
   RECEIVE INDSN('userid.IBIKEYS.LOAD.TEMP')
   ```

   Reply to the restore prompt with

   ```
   DSNAME('userid.IBIKEYS.LOAD')
   ```

6. Using TSO, expand the JCL.TEMP dataset as follows:

   ```
   RECEIVE INDSN('userid.IBIKEYS.JCL.TEMP')
   ```

   Reply to the restore prompt with

   ```
   DSNAME('userid.JCL.DATA')
   ```

7. Make a backup of your FOCLIB.LOAD library.
8. Copy or copy and replace the IBICPUID module from '*userid*.IBIKEYS.LOAD' to your FOCLIB.LOAD library.
9. Create the '*userid*.FOCLIB.LOAD.NEW' output library for the relinked module (R1FNS for Release 7.7.03M and Release 7.7.06M, FOCUS for Release 7.7.09M and higher).
10. Edit the JCLPTF member of the '*userid*.JCL.DATA' library following the instructions included in the JCL, which include:
    - Add a job card at the top of the JCL.
    - Add values to the three SETs at the top of the JCL:
      - PTFLIB = '*userid*.JCL.DATA'
      - OFOCLIB = '*your_production*.FOCLIB.LOAD'
      - NFOCLIB = '*userid*.FOCLIB.LOAD.NEW'

11. Submit the JCLPTF member.
12. Copy and replace the contents of '*userid*.FOCLIB.LOAD.NEW' to your production FOCUS library.
13. Replace the existing FLICENSE file in your production version of FOCUS with the FLICENSE in *userid*.IBIKEYS.FLICENSE.DATA.
14. If you have FOCUS loaded in LPA and/or linklist, you will need to refresh them with this newly zapped module. This is a systems task and should be referred to your systems group.
15. If you are running FOCUS Release 7.7.09M or higher, and are using FOCUSNA, you need to recreate the FOCUSNA module using the newly zapped FOCUS module you have just created.

    The JCL to create the FOCUSNA module is distributed as member FOCUSNA in the FOCCTL.DATA library:

    a. Copy the FOCUSNA member to your FOCUS production JCL library.
    b. Make a copy of your FOCLIB.LOAD library and name it FOCLIBNA.LOAD.
    c. Make the recommended edits to the FOCUSNA JCL. Instructions for editing the JCL for your site are included as comments in the FOCUSNA member.
    d. Execute the FOCUSNA JCL.
    e. Check the FOCLIBNA.LOAD library for the presence of the new FOCUSNA module and EDASAFNA alias.
    f. Replace your production FOCUSNA with the FOCUSNA module just recreated.

       Note: The FOCUSNA module will have an authorization attribute of AC/0, which means that the module was not linked with authorization.
16. Verify your FOCUS license key installation by running FOCUS and issuing the ? CPUID command in both interactive and batch environments. If FOCUS starts, then your installation has been successful. If FOCUS does not start, then the installation has not been successful. Please open a case with Customer Support to review the parameters that have been provided and the process that has been followed.

Previous Licensing Violation Message

If FOCUS is registered and it is run on another processor, the following message display at initialization. This message also displays if there was an error in the License Management Facility installation procedure. FOCUS will not start until it is correctly registered.

```
FOCUSrelease  01/01/2023  13.39.43  27.02
******************************************************
* VIOLATION:
* THIS COPY OF FOCUS IS RUNNING ON AN UNLICENSED CPU.
* PLEASE CONTACT ibi.
* "EX READMEF" FOR COMPLETE DETAILS.
******************************************************
*** Processor mmmm Model nnnn-vv Max pp Site ssss.ss
LICENSED CPU(S):
*** Processor mmmm Model nnnn-vv Max pp > Registration qqqqqqqqqq
>
```

where:

mmmm

Is the Processor ID.

nnnn

Is the model number.

vv

Is the version number.

pp

Is the number of processors on the system.

qqqqqqqqqq

Is the encrypted processor ID.

New Licensing Violation Messages

If your site code is not registered, or the registered site code does not match the installation site code, one of the following messages display at the beginning of all FOCUS

user sessions.

```
FOCUSrelease  01/01/2023  13.39.43  27.02
*********************************************************
* VIOLATION:
* THE SITE CODE FOR YOUR COMPANY WAS NOT REGISTERED
* PROPERLY DURING THE FOCUS INSTALLATION PROCESS.
* PLEASE CONTACT ibi.
* "EX READMEF" FOR COMPLETE DETAILS.
*********************************************************

******************** TOP OF DATA ************************
FOCUSrelease 02/23/2023  14.19.46  xxxx
*********************************************************
* VIOLATION:
* THIS COPY OF FOCUS IS RUNNING ON AN UNLICENSED CPU.
* PLEASE CONTACT ibi.
* "EX READMEF" FOR COMPLETE DETAILS.
*********************************************************
OBSERVED CPU:

******* CEC:  machine type type  model ID model            capacity 45
******* LPAR: name L1                                       capacity 45

******* VM:   name N/A                                      capacity N/A
******* Processor processorinfo >Site xxxx

Licensed Site     xxxx Expires  mmddyyyy
LICENSE PLATES:
16423513B000EBB3400E0119
2981AA172800
FOCUS TERMINATED
******************************** BOTTOM OF DATA ************
```

Verifying ibi FOCUS Installation

The Installation Verification Procedure (IVP) tests the FOCUS installation by creating the CAR database and running tests of several areas of FOCUS processing against it.

In order to run the IVP, you must have a member named CAR in your allocation for DDNAME MASTER. If you do not have the CAR Master File in a data set allocated to DDNAME MASTER, create the member and copy the following syntax into it prior to running the IVP.

```
FILENAME=CAR,SUFFIX=FOC
SEGNAME=ORIGIN,SEGTYPE=S1
 FIELDNAME=COUNTRY,COUNTRY,A10,FIELDTYPE=I,$
SEGNAME=COMP,SEGTYPE=S1,PARENT=ORIGIN
 FIELDNAME=CAR,CARS,A16,$
SEGNAME=CARREC,SEGTYPE=S1,PARENT=COMP
 FIELDNAME=MODEL,MODEL,A24,$
SEGNAME=BODY,SEGTYPE=S1,PARENT=CARREC
 FIELDNAME=BODYTYPE,TYPE,A12,$
 FIELDNAME=SEATS,SEAT,I3,$
 FIELDNAME=DEALER_COST,DCOST,D7,$
 FIELDNAME=RETAIL_COST,RCOST,D7,$
 FIELDNAME=SALES,UNITS,I6,$
SEGNAME=SPECS,SEGTYPE=U,PARENT=BODY
 FIELDNAME=LENGTH,LEN,D5,$
 FIELDNAME=WIDTH,WIDTH,D5,$
 FIELDNAME=HEIGHT,HEIGHT,D5,$
 FIELDNAME=WEIGHT,WEIGHT,D6,$
 FIELDNAME=WHEELBASE,BASE,D6.1,$
 FIELDNAME=FUEL_CAP,FUEL,D6.1,$
 FIELDNAME=BHP,POWER,D6,$
 FIELDNAME=RPM,RPM,I5,$
 FIELDNAME=MPG,MILES,D6,$
 FIELDNAME=ACCEL,SECONDS,D6,$
SEGNAME=WARANT,SEGTYPE=S1,PARENT=COMP
 FIELDNAME=WARRANTY,WARR,A40,$
SEGNAME=EQUIP,SEGTYPE=S1,PARENT=COMP
 FIELDNAME=STANDARD,EQUIP,A40,$
```

By default, the IVP creates a temporary CAR file that is deleted when you exit the FOCUS session. You can retain the CAR database by allocating DDNAME CAR to a permanent physical sequential file with record length and blocksize 4096. The standard naming convention is to use the high-level qualifier for your FOCUS production data sets and the low-level qualifier CAR.FOCUS. After the CAR database is created, allocate it with the disposition SHR in batch or SHR REUSE in TSO.

You can run FOCUS interactively, using a CLIST, or in batch, using JCL. The following sections describe how to run the IVP in each of these environments.

Verifying ibi FOCUS Installation Interactively

If you are running FOCUS interactively, execute the CLIST that was created by the ISETUP installation procedure as member FOCUSC in the *install_hlq*.CONF.DATA data set.

When the FOCUS prompt (>) displays, enter the following command in uppercase, and Press **Enter**.

```
EX CARTEST
```

The following messages display that show that the CAR Master File is valid and that the CAR database was created. The FOC486 message displays if you did not allocate DDNAME CAR to a permanent file.

```
 TEST THE CAR MASTER FOR ERRORS
FOCUS  7.7.06          CREATED 01/17/2018  SITE.ID     XXX
 NUMBER OF ERRORS=     0
 NUMBER OF SEGMENTS=   7  ( REAL=    7  VIRTUAL=   0 )
 NUMBER OF FIELDS=    20  INDEXES=   1  FILES=     1
 TOTAL LENGTH OF ALL FIELDS=  221
OK CAR MASTER
(FOC486) A NEW FILE WAS DYNAMICALLY ALLOCATED
CREATE THE CAR FILE NUMBER OF SEGMENTS TO BE INPUT =102
 LOADING OF DATA COMPLETE
SIMPLE TABLE ON CAR FILE OVERALL RATIO AT END= 1.21
 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press **Enter**.

A basic TABLE request is executed next. The following output displays.

```
 PAGE     1



                TEST1 SIMPLE TABLE

             AVE          AVE

 COUNTRY     RETAIL_COST  DEALER_COST           RATIO
 -------     -----------  -----------           -----
 ENGLAND          11,330        9,463            1.20
 FRANCE            5,610        4,631            1.21
 ITALY            12,766       10,309            1.24
 JAPAN             3,239        2,756            1.18
 W GERMANY         9,247        7,795            1.19

 TOTAL            42,192       34,954            1.21
```

```
 BOTTOM OF PAGE









                                  END OF REPORT
```

Press **Enter** to close the report output.

A test that creates a HOLD file and uses the TABLEF command is executed next. The following output displays.

```
TEST OF A HOLD FILE AND A TABLEF REPORT AFTER
NUMBERS SHOULD BE SAME AS TEST1 WITHOUT RATIO
 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY

 PAGE     1


 TEST2 TABLEF FROM HOLD FILE
             AVE          AVE
 COUNTRY     RETAIL_COST  DEALER_COST
 -------     -----------  -----------
 ENGLAND          11,330        9,463
 FRANCE            5,610        4,631
 ITALY            12,766       10,309
 JAPAN             3,239        2,756
 W GERMANY         9,247        7,795

DEFINE TEST WITH REPORT .. TOTAL MARGIN SHOULD=99.50
FILE NAME            FIELD NAME                        FORMAT  SEGMENT
TYPE
CAR                  MARGIN                            D10.2         4
 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press **Enter** to close the report output.

A test of the DEFINE command is executed next. The following output displays.

```
PAGE     1
```

```
TEST3 DEFINE MARGIN

TOTAL SUM OF MARGIN =        99.50

MODEL                           MARGIN

_____                           ______

INTERCEPTOR III                  19.48

TR7                              18.83

V12XKE AUTO                      19.54

XJ12L AUTO                       20.52

504 4 DOOR                       21.14


TOTAL                            99.50









                                END OF REPORT
```

Press **Enter** to close the report output.

A test of the SCAN command is executed next. The following output displays.

```
SCAN TEST  SEATS 2 CHANGED TO SEATS 4

COUNTRY=ENGLAND    CAR=JAGUAR           MODEL=V12XKE AUTO
SEAT=  2
COUNTRY=ENGLAND    CAR=JAGUAR           MODEL=V12XKE AUTO
SEAT=  4
TESTING SML WITH SHORT MODEL

 PAUSE.. PLEASE ISSUE CARRIAGE RETURN WHEN READY


PAGE     1


            AVE
            RETAIL_COST
            -----------
MILAGE RANGE
0 TO 12          16,495
13 TO 20         11,069
COMPARISON            1









                                END OF REPORT
```

Press **Enter** to close the report output.

A GRAPH request is executed next. The following output displays.

```
TESTING GRAPHS
```

```
18 RECORDS SHOULD BE RETRIEVED AND PLOTTED

 PAUSE..PLEASE ISSUE CARRIAGE RETURN WHEN READY



  R     40,000 +
  E            I
  T            I
  A            I
*
  I            I                                                  .
  L            I                                             .
  _     20,000 +                                       .
  C            I                             .  *
  O            I                  . * ** . .
  S            I           . * *
  T            I   . *****
               I *
             0 +----+----+----+----+----+----+----+----+----+----+----+-
---+

              2,000     6,000    10,000    14,000    18,000    22,000
26,000
                   4,000     8,000    12,000    16,000    20,000
24,000

                                       DEALER_COST

 AT END OF TEST MAJOR AREAS OK

 >  >
```

```
```

These tests show that FOCUS is installed and operational. Enter the following command to exit FOCUS.

```
FIN
```

Verifying ibi FOCUS Installation in Batch

If you are running FOCUS in batch, edit the FOCUS JCL that was created by ISETUP as member FOCUS in the *install_hlq*.CONF.DATA data set to add the installation verification procedure.

The installation verification procedure will be added in the allocation for DDNAME SYSIN. The FOCUS member in the *install_hlq*.CONF.DATA data set has the following JCL statements.

```
//SYSIN    DD *
Put your input here in uppercase
FIN
```

Replace the line that says **Put your input here in uppercase** with the following command.

```
EX CARTEST
```

Edit the JOB card with your information and submit the job. Output similar to the following will appear in the SYSPRINT DDNAME of the FOCUS step. Note that the FOC486 message will display if you did not allocate DDNAME CAR to a permanent file.

```
FOCUS  7.7.06  11.16.57  07/27/2018  CARTEST   LINE   20  XXX

 SET MSG=OFF
FOCUS  7.7.06          CREATED 01/17/2018  SITE.ID     XXX
 NUMBER OF ERRORS=     0
 NUMBER OF SEGMENTS=   7  ( REAL=    7  VIRTUAL=   0 )
 NUMBER OF FIELDS=    20  INDEXES=   1  FILES=     1
 TOTAL LENGTH OF ALL FIELDS=  221
OK CAR MASTER
```

```
(FOC486) A NEW FILE WAS DYNAMICALLY ALLOCATED
CREATE THE CAR FILE NUMBER OF SEGMENTS TO BE INPUT =102
 LOADING OF DATA COMPLETE
SIMPLE TABLE ON CAR FILE OVERALL RATIO AT END= 1.21
 PAGE     1
               TEST1 SIMPLE TABLE
            AVE          AVE
COUNTRY     RETAIL_COST  DEALER_COST           RATIO
-------     -----------  -----------           -----
ENGLAND          11,330        9,463            1.20
FRANCE            5,610        4,631            1.21
ITALY            12,766       10,309            1.24
JAPAN             3,239        2,756            1.18
W GERMANY         9,247        7,795            1.19

TOTAL            42,192       34,954            1.21

BOTTOM OF PAGE
```

```
TEST OF A HOLD FILE AND A TABLEF REPORT AFTER

NUMBERS SHOULD BE SAME AS TEST1 WITHOUT RATIO

 PAGE     1



TEST2 TABLEF FROM HOLD FILE
            AVE          AVE
COUNTRY     RETAIL_COST  DEALER_COST
-------     -----------  -----------
ENGLAND          11,330        9,463
FRANCE            5,610        4,631
ITALY            12,766       10,309
JAPAN             3,239        2,756
W GERMANY         9,247        7,795
```

```
DEFINE TEST WITH REPORT .. TOTAL MARGIN SHOULD=99.50

FILE NAME            FIELD NAME                      FORMAT  SEGMENT
TYPE
```

```
CAR                  MARGIN                          D10.2         4
 PAGE     1

 TEST3 DEFINE MARGIN
 TOTAL SUM OF MARGIN =        99.50
 MODEL                          MARGIN
 -----                          ------
 INTERCEPTOR III                 19.48
 TR7                             18.83
 V12XKE AUTO                     19.54
 XJ12L AUTO                      20.52
 504 4 DOOR                      21.14

 TOTAL                           99.50
```

```
SCAN TEST  SEATS 2 CHANGED TO SEATS 4

COUNTRY=ENGLAND    CAR=JAGUAR           MODEL=V12XKE AUTO
SEAT=  2
COUNTRY=ENGLAND    CAR=JAGUAR           MODEL=V12XKE AUTO
SEAT=  4
TESTING SML WITH SHORT MODEL

 PAGE     1




             AVE

             RETAIL_COST

             -----------

 MILAGE RANGE

 0 TO 12          16,495

 13 TO 20         11,069

 COMPARISON            1
```

```
TESTING GRAPHS
18 RECORDS SHOULD BE RETRIEVED AND PLOTTED


       35,000 +
              I
              I
              I                                                        *
       30,000 +                                                     .
              I                                                  .
              I                                               .
              I                                            .
              I                                         .
 R     25,000 +                                      .
 E            I                                   .
 T            I                                .
 A            I                             .
 I     20,000 +                          .
 L            I                       .
 _            I                    *
 C            I                 .
 O     15,000 +              .
 S            I          * **
 T            I          .
              I       .
       10,000 +       *
              I     * *
              I   .
              I  * *
        5,000 +  **
              **
              I
              I
            0 +-----+-----+-----+-----+-----+-----+-----+-----+-----+

              2,500       7,500      12,500      17,500      22,500
                    5,000      10,000      15,000      20,000      25,000

                                    DEALER_COST


AT END OF TEST MAJOR AREAS OK
```

This output shows that FOCUS is installed and operational.

Next Steps

In the next steps, you will customize your CLIST or JOB to add private data sets and to move FOCUS to production. After performing those steps, you can run the IVP again to make sure the customized version of FOCUS is still operational. If you run into any problems at that stage, review your customization steps.

You may have inadvertently:

- Renamed required libraries so that they are not allocated correctly.
- Erased the aliases so that the required name transformations cannot be completed correctly.
- Moved members in the libraries so that they cannot be found using the standard DDNAMEs

Specifying Non-Standard Data Set Names

ISETUP asks you for the high-level qualifier for your current production version of FOCUS. Based on this qualifier and a list of all of the standard data set names (low level qualifiers), ISETUP installs all the necessary FOCUS libraries. The list of low level qualifiers ISETUP uses is in member FOCSNAME in the F.HOME.DATA data set:

- If you use non-standard low level qualifiers for your FOCUS data sets, you must edit this file. Once you have done this, ISETUP will use these qualifiers for future installations, as well as the current one.
- If you use multiple prefixes for your FOCUS data sets, you can edit this file to specify the correct data set names.

Important: You must use the version of FOCSNAME provided for the current release, not the version from a previous installation.

If you use only the standard FOCUS data set names, you do not have to perform this step.

Note: Only ISETUP uses FOCSNAME. FOCUS, once installed, does not use this member.

To edit FOCSNAME, open member FOCSNAME in the F.HOME.DATA data set and follow the instructions entered as comments at the top of the file.

The following is an example of a partial listing of the FOCSNAME member. The contents of this member may change over time, but should be simliar to the following.

```
//*********************************************************************
//*Purpose: To Allow the setting of site specific low level qualifiers*
//*         for z/OS PDS FOCUS install process                        *
//*                                                                   *
//*         Change the values below for the different datasets that   *
//*         may be allocated by the installation process. Keep in     *
//*         mind that a "qualifier" will be used as a suffix to these *
//*         names so be careful as to length.                         *
//*                                                                   *
//*********************************************************************
//         SET  EDALOAD='F.HOME.LOAD'   Load Library
//         SET  EDAERR='F.HOME.ERR'     Errors files
//         SET  EDAHMAS='F.HOME.MAS'    Master files
//         SET  EDAHFEX='F.HOME.FEX'    RPCs
//         SET  EDAHETC='F.HOME.ETC'    Html and Text files
//         SET  EDAHACX='F.HOME.ACX'    Access files
//         SET  EDAHBIN='F.HOME.BIN'    Binary files
//         SET  EDAHDAT='F.HOME.DATA'     Isetup library
//         SET  PDSFOCD='CONF.DATA'     Focus Install/Runtime JCLs
//         SET  EDACCFG=&CONFTYPE..CFG  Configuration files
//         SET  EDAPROF=&CONFTYPE..PRF
//         SET  EDACACX=&CONFTYPE..ACX
//         SET  EDACMAS=&CONFTYPE..MAS
//         SET  EDACSQL=&CONFTYPE..SQL
//         SET  FOCUSSU=&CONFTYPE..FOCUSSU.FOCUS
```

As described, where necessary overwrite each standard name with your non-standard name.

For example, if you use the name CONFIG.DATA for your production configuration library, change the following line:

```
// SET  EDACCFG=&CONFTYPE..CFG        Configuration files
```

It should now say:

```
// SET  EDACCFG=&CONFTYPE..CONFIG.DATA       Configuration files
```

Creating a CLIST or Batch Job to Invoke the Installed Version of ibi FOCUS

Edit a copy of the CLIST or batch JCL you use to invoke your current production version of FOCUS. Create a new CLIST or batch job with the new high level qualifiers to invoke the upgraded version of FOCUS.

The fact that the FOCUS data sets are distributed as PDSEs requires you to concatenate these PDSE libraries in front of any fixed blocked private data sets you want to carry forward from prior releases.

When you invoke FOCUS, the banner indicates that the version was upgraded.

Generating Sample ibi FOCUS Data Sources

The sample FOCUS data sources are delivered as tutorials in a jtar file under USS, which must be unloaded to an application on z/OS.

Generate Sample ibi FOCUS Data Sources

Procedure

1. From the TSO command line, enter the following command.

   ```
   OMVS
   ```

   This command puts you in your root directory under USS. We will refer to this directory as /u/user1.

2. Create a directory for the jtar file and open that directory.

   The following commands create a directory under your root directory named *unload* and opens that directory.

   ```
   mkdir unload
   cd unload
   ```

3. FTP to your MVS host.

   The following command opens FTP to a host named *mvshost*.

   ```
   ftp mvshost
   ```

   You will be prompted to enter your user ID and password.

4. Establish binary transfer mode, transfer the jtar file to the directory you created, and quit out of FTP.

   Issue the following commands.

   ```
   bin
   get 'hlq.f.home.bin(jtar)' jtar
   quit
   ```

   where *hlq* is the high-level qualifier under which you installed FOCUS.

5. Unload the jtar file.

   Issue the following command.

   ```
   tar -xvf jtar
   ```

6. Exit from USS.

7. Create a PDSE with a member named EDASERVE and concatenate it to the data sets allocated to DDNAME ERRORS in your FOCUS JCL or CLIST.

   The EDASERVE member enables APPS by setting the APPROOT variable, and sets the edahome_dir variable to the directory you created under USS. For example:

   ```
   APPROOT=USER1.APPS
   edahome_dir=/u/user1/unload
   ```

   Note: The edahome_dir variable line must be in lowercase.

8. Start FOCUS and issue the following command to create the FOCUS data sources.

   ```
   EX SAMPLTUT TUTORIAL=LEGACY
   ```

   The FOCUS data sources will be generated in an application named IBISAMP, which will consist of a set of data sets whose high-level qualifier consists the value of

APPROOT, and the application name. For example, the following is a partial list of the data sets created for APP IBISAMP, where the user ID is USER1 and APPROOT is set to USER1.APPS:

```
USER1.APPS.IBISAMP.ACCESS.DATA
USER1.APPS.IBISAMP.BROKERS.FOCUS
USER1.APPS.IBISAMP.CAR.FOCUS
USER1.APPS.IBISAMP.CASHFLOW.FOCUS
USER1.APPS.IBISAMP.COURSE.FOCUS
USER1.APPS.IBISAMP.DTD.DATA
USER1.APPS.IBISAMP.EDUCFILE.FOCUS
USER1.APPS.IBISAMP.EMPDATA.FOCUS
USER1.APPS.IBISAMP.EMPLOYEE.FOCUS
```

If you do not want to run focus using APPS, you can copy the application data sets to other data sets and remove the data set you created with the EDASERVE member from the ERRORS concatenation.

For more information about APPS, see the *ibi™ FOCUS® Developing Applications* manual.

Moving the Installed Version of ibi FOCUS to Production After Testing

After successful testing of the service pack or upgrade, move the test environment to production according to the standards of your site.

Introducing Talk Technology

Talk Technology consists of a set of window-driven products that you can use to interactively create reports, produce graph requests, write a FOCUS data description (Master File), and generate procedures (FOCEXECs) to manage your data.

Talk Technology is easy to use. All you need to do is select options and answer simple questions on your terminal. Talk Technology then automatically translates your specifications into native FOCUS syntax.

The four Talk Technology products are:

- TableTalk
- PlotTalk
- FileTalk
- ModifyTalk

They are productivity tools for the whole FOCUS community. With them, new users have facilities that automate the whole cycle of designing (FileTalk), maintaining (ModifyTalk), reporting (TableTalk), and graphing (PlotTalk) data from their own file systems.

As you read this content, you will notice the use of the word *view*. A view is a group of related fields that may be accessed by a single FOCUS request, such as TABLE or GRAPH.

A view may consist of:

- A single FOCUS database or a relational table.
- A combination of tables and/or databases created using a JOIN command or an embedded cross-reference.
- A subset of fields. This option depends on defined security restrictions.

Views which require a JOIN need to be defined before entering Talk Technology. For more information on defining joined databases and the automated JOIN facility, see the *FOCUS for IBM Mainframe User's Manual*.

TableTalk

FOCUS TableTalk is a window-driven facility for preparing report requests. TableTalk is designed to create report requests quickly without the worry of debugging them. You can execute them immediately when they are complete. Because TableTalk is designed to offer flexibility without introducing inaccuracy, it also serves as an ideal educational tool. You learn about the FOCUS report writing language as you use this facility to generate your report requests.

TableTalk enables you to design and write your own report requests quickly when you choose options from its prompt windows. When your requests are complete, you also have options to run, revise, save, and clear your work.

PlotTalk

FOCUS PlotTalk is a window-driven facility for creating graphs using data from your databases. PlotTalk guides you through the process of building graph requests in FOCUS syntax.

PlotTalk assembles a graph request for you as you make selections from pop-up lists on prompt windows. Like the other Talk Technologies described in this content, PlotTalk only permits selections of valid options from within the FOCUS GRAPH environment. This ensures that your requests will not contain syntax errors, and makes PlotTalk an excellent tool for learning GRAPH command syntax.

FileTalk

FOCUS FileTalk is a window-driven facility for building new FOCUS databases. FileTalk is designed to create Master Files quickly while preventing you from making errors. You can add data to the file immediately after defining your last field. All you have to do is plan the structure of your file before you begin your FileTalk session.

FileTalk enables you to create, design, and add data to your FOCUS databases using AUTOMOD when you choose options from its prompt windows. When your Master File is complete, you can also generate a diagram of your FOCUS database to help you check the structure of the file you have just created. Simply select an option on the pop-up lists in the prompt windows, and press Enter.

ModifyTalk

FOCUS ModifyTalk is a window-driven facility for generating full-screen, function key-oriented MODIFY procedures for single-segment or multi-segment files. ModifyTalk is designed to create MODIFY procedures quickly without the worry of debugging them. You can execute the MODIFY procedures immediately after completing your selections. All you have to do is plan the actions you want to take when maintaining your FOCUS databases.

You can use ModifyTalk to develop FOCEXECs that display, update, add, and delete records in your FOCUS files. You simply choose the file segments to be altered and select the data management actions to be taken. ModifyTalk automatically generates all the screens and processing logic for your new database maintenance procedure.

Alternate Ways to Enter Talk Technology Products

You may access Talk Technology products from the FOCUS Menu and Toolkit window-driven facilities.

ibi FOCUS Menu

To use the FOCUS Menu to access any of the four Talk products, invoke the Menu first from your FOCUS session in MVS. Type the following command and press Enter:

```
EX FMMAIN
```

Then, continue:

- To access TableTalk or PlotTalk, select the *TableTalk* or *PlotTalk* option on the Menu's first screen and press Enter.

  The initial screen for either TableTalk or PlotTalk appears. From this point, continue

with the instructions in TableTalk or PlotTalk, respectively.

- To access FileTalk or ModifyTalk, select the *Next* option on the Menu's first screen. On the next screen, select the *FileTalk* or *ModifyTalk* option and press Enter.

  The initial screen for either FileTalk or ModifyTalk appears. From this point, continue with the instructions in FileTalk or ModifyTalk, respectively.

ibi FOCUS Toolkit

To use the FOCUS Toolkit to access any of the Talk Technology products, except FileTalk, invoke the Toolkit first from your FOCUS session in MVS. Type the following command and press Enter:

```
EX ISHFSHLL
```

Then, continue:

- To access TableTalk:

  1. Select the *Reporting Facilities* option and press Enter.
  2. You will be prompted to select a file for reporting. Scroll down the list with PF8, select your file, and press Enter.
  3. Select the *Menu Driven Report Writer* option and press Enter.

  The initial screen for TableTalk appears. From this point, continue with the instructions in TableTalk.

- To access PlotTalk:

  1. Select the *Decision Support Selections* option and press Enter.
  2. You may be prompted to select a file, if you have not already. Scroll down the list with PF8, select your file, and press Enter.
  3. Select the *Menu Driven Graphics Creator* option and press Enter.

  The initial screen for PlotTalk appears. From this point, continue with the instructions in PlotTalk.

- To access ModifyTalk:

  1. Select the *Maintain Data* option and press Enter.

2. You may be prompted to select a file, if you have not already. Scroll down the list with PF8, select your file, and press Enter.
3. Select the *Maintain a File* option and press Enter.

The initial screen for ModifyTalk appears. From this point, continue with the instructions in ModifyTalk.

Using Talk Technology Prompt Windows and Special Keys

All Talk Technology products use windows to prompt for or display option lists to gather information, and identical program function (PF) and arrow keys to manipulate these windows. The following subsections describe options and facilities that enable you to move through a Talk Technology session.

Prompt Windows

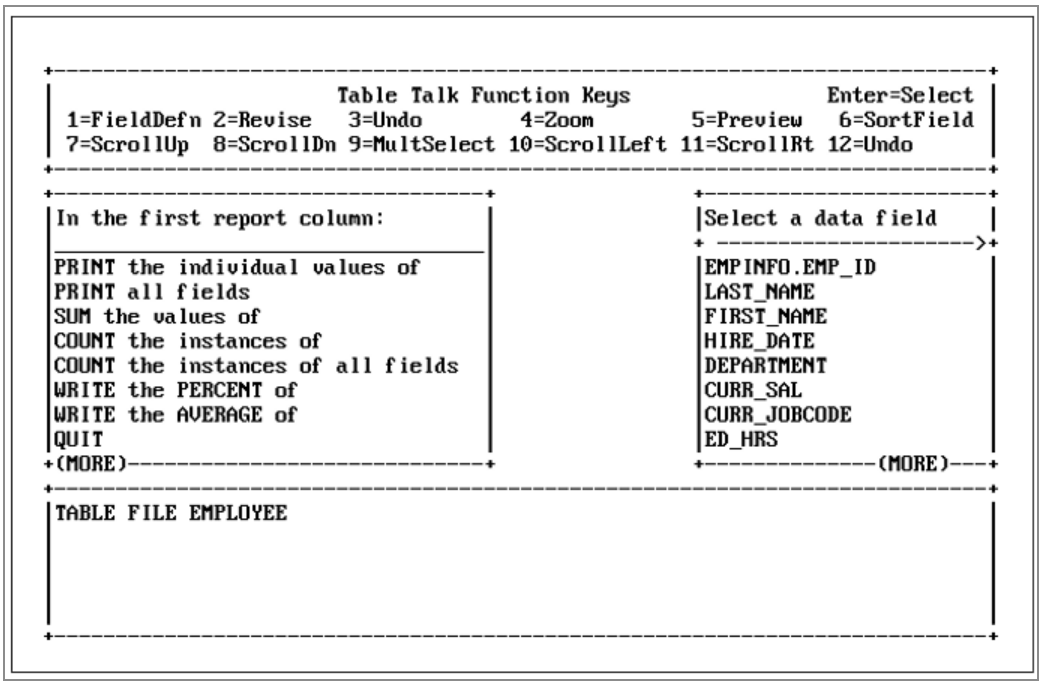
Talk Technology guides you through the development process with windows that prompt you for input. Talk Technology screens usually display more than one window. The highlighted window where the cursor is positioned is the active window, the window awaiting your selection. You may have to adjust your screen display to see the highlighting effect.

Some windows that have lists enable you to select multiple options before pressing Enter. This streamlines your whole Talk Technology session by cutting down on the number of steps and the amount of time it takes to define multiple parameters to the Talk Technology facility.

Other windows require nothing more of you than moving the cursor (using the up and down arrow keys) to the appropriate option and pressing Enter, while some windows require that you type some information.

Note: Talk Technology products convert all input into uppercase text.

Consider the following window. Note that only one window is highlighted, the *Select a data field* window:



There are three ways you can select items from this window:

- Move the cursor to your selection with the up and down arrow keys and press Enter.
- Move the cursor to your selection by typing the first letter(s) of that option and pressing Enter. In cases where there are many options that begin with the same letter, the cursor moves to the first option beginning with the letter you pressed. Press the same letter again, and the cursor moves to the next option that begins with the letter you typed and so on. At the end of the list, the cursor searches from the top again and press Enter to select.
- Select several options with PF9 before pressing Enter. This is called the Multi-Select feature. It reduces the number of steps needed to specify several parameters. Simply move the cursor to your first choice and press PF9. Then, move the cursor to your second choice and press PF9 again. When the cursor is on your last choice press Enter to process them all.

  Note: Multi-Select is not available with PlotTalk or FileTalk.

To process your selection and move the cursor to the next window, press Enter. Note that the next window immediately begins where the last window ends, thereby propelling you through the entire Talk session quickly and efficiently.

If the list of options is too long to fit in the prompt window, you will need to scroll to see all your choices. The word MORE tells you that the options continue beyond the current window, and depending on its location, in which direction you scroll to see the remaining

options. If MORE appears at the bottom of the list, you scroll down. If MORE appears at the top of the list, you scroll up.

After determining which direction to scroll, use the keys described in the following table to move to your selections.

Key	Action
Arrow Keys	Moves the cursor one selection at a time in the specified direction.
Tab	Moves the cursor to the next item in the selection list (in some windows).
Backspace	Moves the cursor to the preceding item in the selection list (in some windows).
PF3	Removes your last selection and returns you to the previous TableTalk, PlotTalk, or FileTalk prompt window.
PF7	Scrolls up a list of items within a window one screen at a time.
PF8	Scrolls down a list of items within a window one screen at a time.
PF10	Moves one screen width left within the fieldlist windows in TableTalk and PlotTalk.
PF11	Moves one screen width right within the fieldlist windows in TableTalk and PlotTalk.

Talk Technology PF Keys

Talk Technology PF key assignments enable quick access to certain commands and functions. The keys appear at the top of all Talk Technology screens and are explained in the following table.

Key	Action
PF1	TableTalk and PlotTalk — displays available field definitions. To display a field

Key	Action
	definition, your cursor must be in a window that displays fields. Not applicable to FileTalk or ModifyTalk.
PF2	TableTalk and PlotTalk — activates Revise Mode, which enables you to edit your request. Pressing the key again returns you to your current Talk session. FileTalk and ModifyTalk — expands the bottom window to a full screen for reviewing the Master File in FileTalk or the procedure comments in ModifyTalk. Pressing the key again returns you to your current Talk session.
PF3	TableTalk, PlotTalk, and FileTalk — erases the last selection and its corresponding FOCUS syntax in the bottom window, and returns you to the previous window. ModifyTalk — Quits and returns you to the FOCUS prompt.
PF4	TableTalk and PlotTalk only — activates the ZOOM feature which enables you to view up to 60 fields in the *Select a data field* window and returns you to the original screen when pressed again.
PF5	TableTalk only — displays a preview of a report at any time during a session.
PF6	TableTalk and PlotTalk only — sorts fields while *Select a data field* is activated in database order, alphabetic order, or qualified by segment. To activate the sort field option, SET FIELDNAME must be set to NEW. ModifyTalk only — cancels all choices for your current segment and returns to the first selection of the preceding segment.
PF7	Scrolls one screen up to the beginning of a selection list.
PF8	Scrolls one screen down to the end of a selection list.
PF9	TableTalk only — selects more than one item from a single window, when appropriate. ModifyTalk only — selects fields for update, after you have selected the option to update/view a segment and a list of fields in that segment is displayed.

Key	Action
PF10	TableTalk and PlotTalk only — scrolls left within the fieldlist window (see PF4 and PF6 above). ModifyTalk only — selects fields for view, after you have selected the option to update/view a segment and a list of fields in that segment is displayed.
PF11	TableTalk and PlotTalk only — scrolls right within the fieldlist window.
PF12	TableTalk, PlotTalk, and FileTalk — erases the last selection and its corresponding FOCUS syntax in the bottom window, and returns you to the previous window. ModifyTalk — erases last selection.
Enter	Records the selection where the cursor is positioned.

Note: These keys and their assignments are a permanent part of Talk Technology. You cannot change the assignments for these keys, or assign functions to unused keys.

Using the Samples: Requirements

Before you can generate a report, graph, or MODIFY procedure using TableTalk, PlotTalk, or ModifyTalk, respectively, you must have:

1. A data file. This may be a FOCUS file or a non-FOCUS data structure supported by a FOCUS interface.

   The sample requests in this manual use the FOCUS EMPLOYEE data file available with the FOCUS installation.

2. A Master File for the data file you want to access. If you do not have a Master File, you can create one with FileTalk, the FOCUS text editor, TED, or your system editor. **For more information on Master File, see Appendix A, Overview of the Master File Description.**

   The Master File for the EMPLOYEE data file is member EMPLOYEE in a partitioned dataset (PDS) allocated to the ddname MASTER, under MVS.

   If you are working with non-FOCUS data files, they must have Master Files that meet FOCUS standards for external files, and you must be running the appropriate

interface for that external file.

You must identify non-FOCUS files with the ALLOCATE (TSO) or DYNAM (TSO and MSO) command to make them accessible through TableTalk. For more information on ALLOCATE and DYNAM, see the *FOCUS for IBM Mainframe User's Manual*.

Under MVS, the search covers only the partitioned datasets allocated to the ddname MASTER

Note: You may have access to a Master File without having access to its corresponding data file. To retrieve or modify data in FOCUS, you must have access to both. Talk Technology facilities enable you to enter and attempt to execute a request, then indicate that the data is missing. To avoid this situation, consult your FOCUS administrator for information on which data files you can access.

Creating the Required Databases

Before you generate the sample requests in this manual, you must first invoke FOCUS then at the FOCUS prompt (>), create the database you will be using. Once you do this, you can use any of the Talk Technologies described, and in any order you wish.

To create the EMPLOYEE database:

- Under MVS, issue the command:

```
EX EMPTSO
```

- To create the SALES database under MVS, issue the command:

```
EX GSTART
```

To resolve any problems you may have, see your FOCUS system administrator.

Identifying the Files You Need

The samples in this manual use the EMPLOYEE and SALES databases. But you can also use your own databases with these Talk Technologies.

To demonstrate how to use your own database, we use a hypothetical database named MYDATA as an example. To use either TableTalk or PlotTalk, you need, in addition to the MYDATA database, a Master File for MYDATA.

Under TSO and MSO, you must have a PDS allocated to ddname MASTER and it must have the member MYDATA. You must also allocate the MYDATA database to the ddname MYDATA. If you have any difficulty, see your FOCUS administrator.

TableTalk

This chapter describes how to create simple report requests using TableTalk. By working through the tutorial, you will learn how to generate a request that produces a report containing specific data items using sorting and selection criteria. This example will show the identification number, last name, and first name of all employees in each department who earn an annual salary greater than $25,000.

Many TableTalk options are not used in creating the sample request. After you have created the sample, try using all of the options to see how they work.

TableTalk Requirements

In order to use TableTalk, you must have a Master File and the data file it describes. See Introducing Talk Technology for a discussion of these requirements and steps used to create a data file.

In TableTalk, if you have access to a Master File without having access to its corresponding data file, you will not realize the problem until after you have entered and tried to execute a request. At that point, FOCUS displays an error message:

```
(FOC036) NO DATA FOUND FOR FOCUS FILE NAMED: filename
```

To avoid this situation, consult your FOCUS administrator for information on which data files you can access.

Report requests created with TableTalk are limited to 24 lines. When you reach the 24-line limit, you can execute or save the request, among other options discussed later in this content. Report requests created or edited outside of TableTalk have no line limitation.

Talk Technology products convert all input into uppercase text. In TableTalk, this may affect computations, column titles, record selection tests, headings, and footings.

Basic Elements of a Report Request

The basic elements of a report request are:

- Report columns
- Sorting options
- Selection criteria

The report columns in this content's sample request are:

- Employee identification number
- Employee first name
- Employee last name

The sorting option in this sample request is the DEPARTMENT field in the EMPLOYEE data file because the report organizes all the information by department.

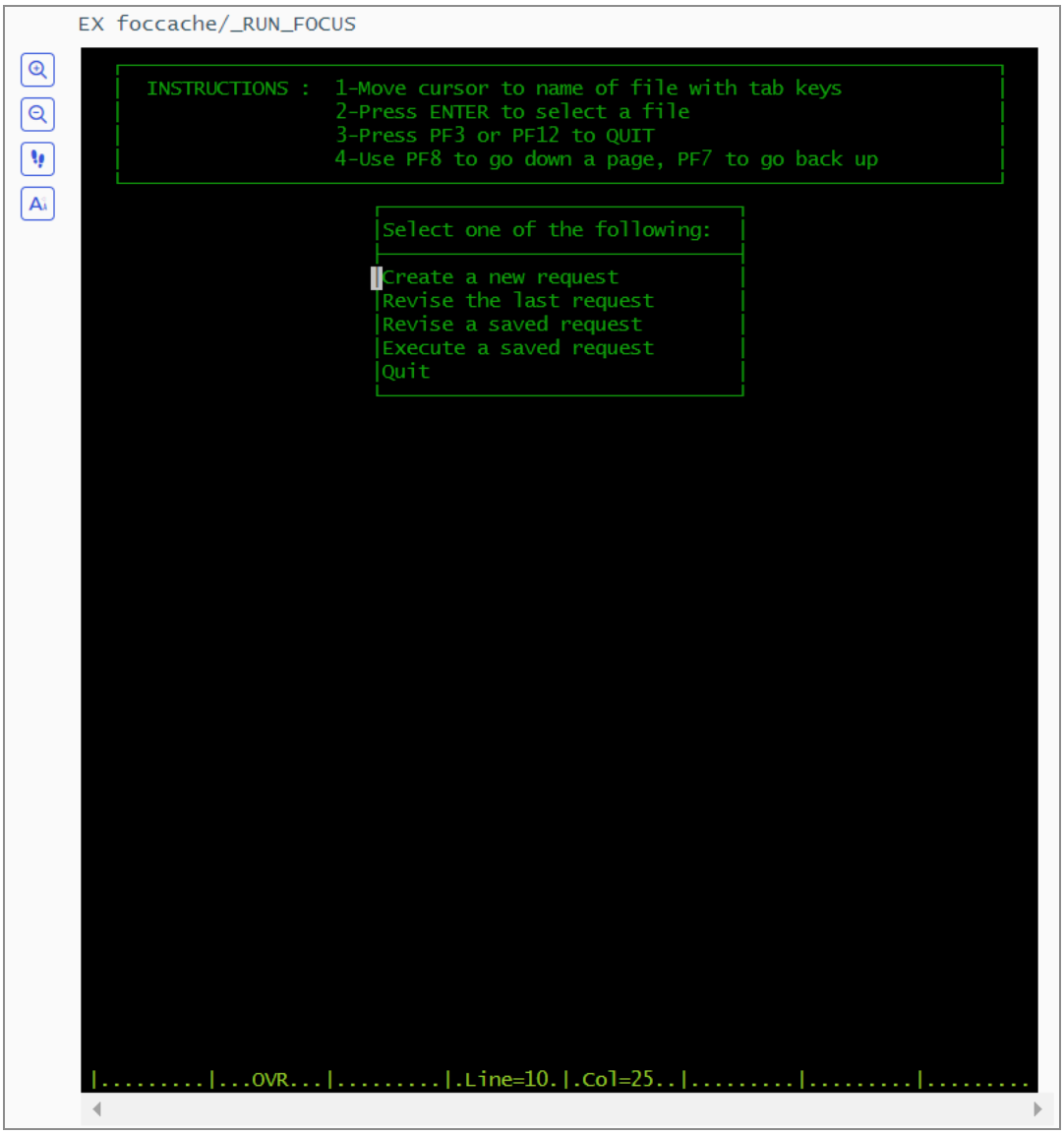
The selection criterion in this request extracts data for all employees in each department with salaries greater than $25,000 a year. Therefore, when the completed request is executed, the resulting report should display the identification numbers, first names, and last names of all employees in each department that have annual salaries greater than $25,000.

Entering TableTalk

There are several ways you can invoke TableTalk. The first way to invoke TableTalk is to type the following command at the FOCUS command prompt and press Enter.

```
TABLETALK
```

The first screen that you see is the TableTalk *Select one of the following* window, as shown in the following image:

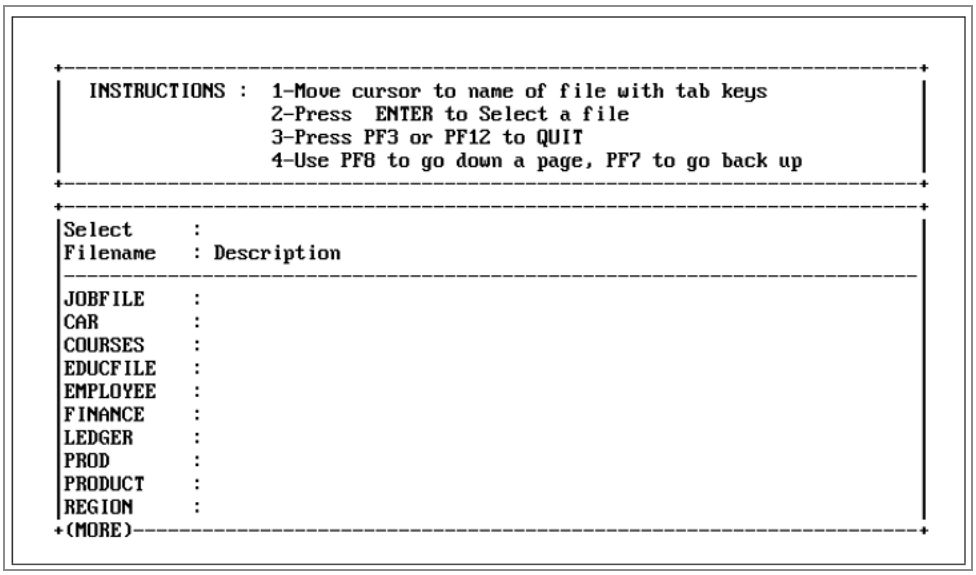


Select the option

```
Create a new request
```

The next screen that you see is the TableTalk *File Selection* window. Notice that the screen displays the filename and a description of the file, if one exists. The screen displays an additional column for the application name. When the file list screen is displayed, TableTalk positions the cursor on the first file in the alphabetical list. If you have already used TableTalk or PlotTalk to generate a report or graph in the current FOCUS session, the last file you used is listed first. This screen lists FOCUS files or views that are currently available for reporting in an alphabetical list.

Your screen may display different files from the following:



To select the EMPLOYEE file, use a combination of the PF7 (page up), PF8 (page down), arrow, and TAB keys to position the cursor next to EMPLOYEE and press Enter. TableTalk automatically generates the following FOCUS code in the bottom window of the screen that follows:

```
TABLE FILE EMPLOYEE
```

Alternate Ways to Enter TableTalk

You can also invoke TableTalk by typing the following command at the FOCUS command prompt and pressing Enter:

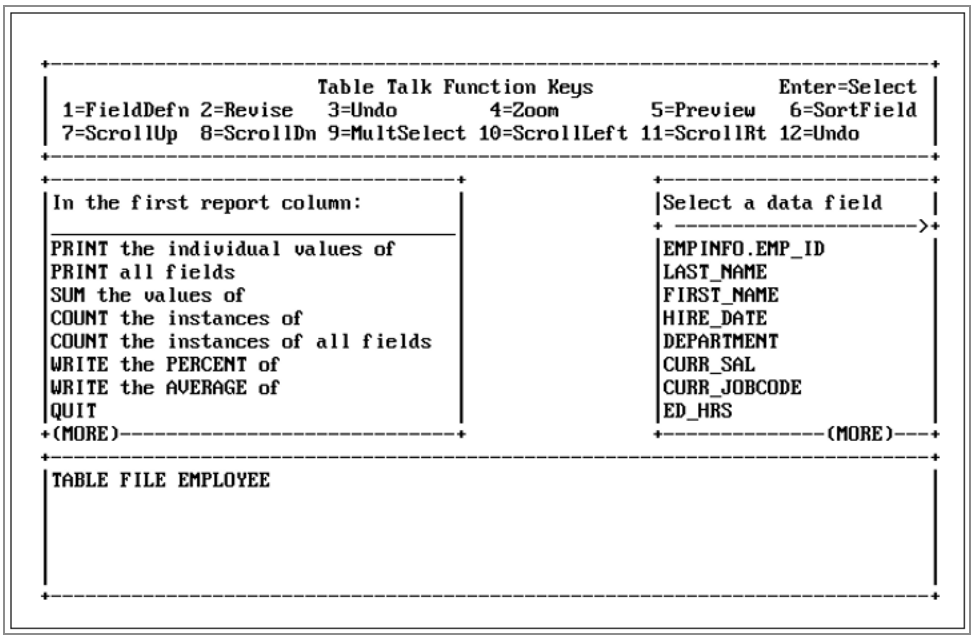
```
TABLETALK FILE filename
```

Where *filename* is the name of the file or view you want to use. This command enables you to bypass the File Selection window.

Other ways to enter TableTalk include selecting it from the FOCUS Menu facility or the FOCUS Toolkit facility. Instructions for using both facilities are provided in Introducing Talk Technology.

Aborting a Request

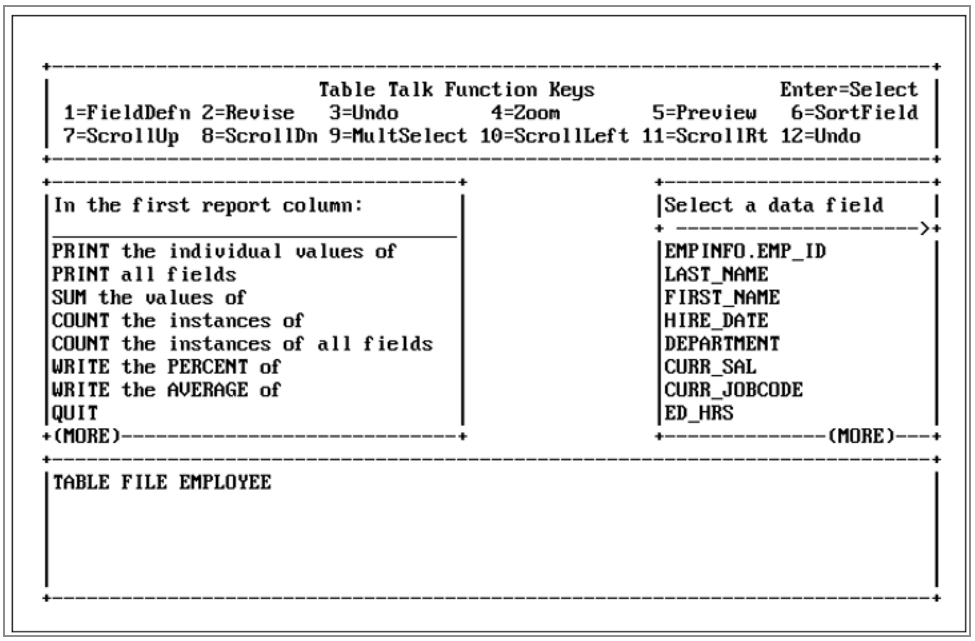
You can abort a request and return to the FOCUS command prompt at any point in TableTalk. Simply press PF3 until you reach the *In the first report column* window, as shown in the following image.



To exit TableTalk and return to the FOCUS command prompt, select the *QUIT* option. The *QUIT* option is also available on the last TableTalk window.

Selecting Report Columns

After you select a file, in this case EMPLOYEE, TableTalk displays the following screen:



This screen, like all screens in TableTalk, is divided into three main areas:

- Special key information is provided in the top window. For more information on special keys, see Introducing Talk Technology.
- Selection windows appear in the middle.
- Report requests being created are displayed in the bottom window.

The first window is the *In the first report column*: window. You can choose to print or count the instances of all fields from the first action menu. All fields in the selected file are printed or counted. If the Master File is a multi-path file, only one path is printed. There is a maximum of 256 fields that may be printed in a single request.

The following table describes the options that are available to you in the *In the first report column:* window:

Option	Description
`PRINT the individual values of`	Lists the values of the field that you specify in your report request. No summation is performed.

Option	Description
`PRINT all fields`	Prints all fields in a single path file and all fields in the left path of a multi-path file.
`SUM the values of`	Adds the values of the field that you specify in your report request. Only one summary line is printed.
`COUNT the instances of`	Counts the number of instances that exist for a specified field. COUNT counts the instances of data in a file, not the distinct values.
`COUNT the instances of all fields`	Counts the occurrences of all fields in a single path file and all fields in the left path of a multi-path file.
`WRITE the PERCENT of`	Computes the percentage from the total values for a specified field in a sort group.
`WRITE the AVERAGE of`	Sums the field values within a sort group divided by the number of records in that sort group.
`WRITE the MAXIMUM of`	Sums the maximum field values within a sort group.
`WRITE the MINIMUM of`	Sums the minimum field values within a sort group.
`PRINT the COMPUTED`	Lists the computed field values that you specify in your report request.

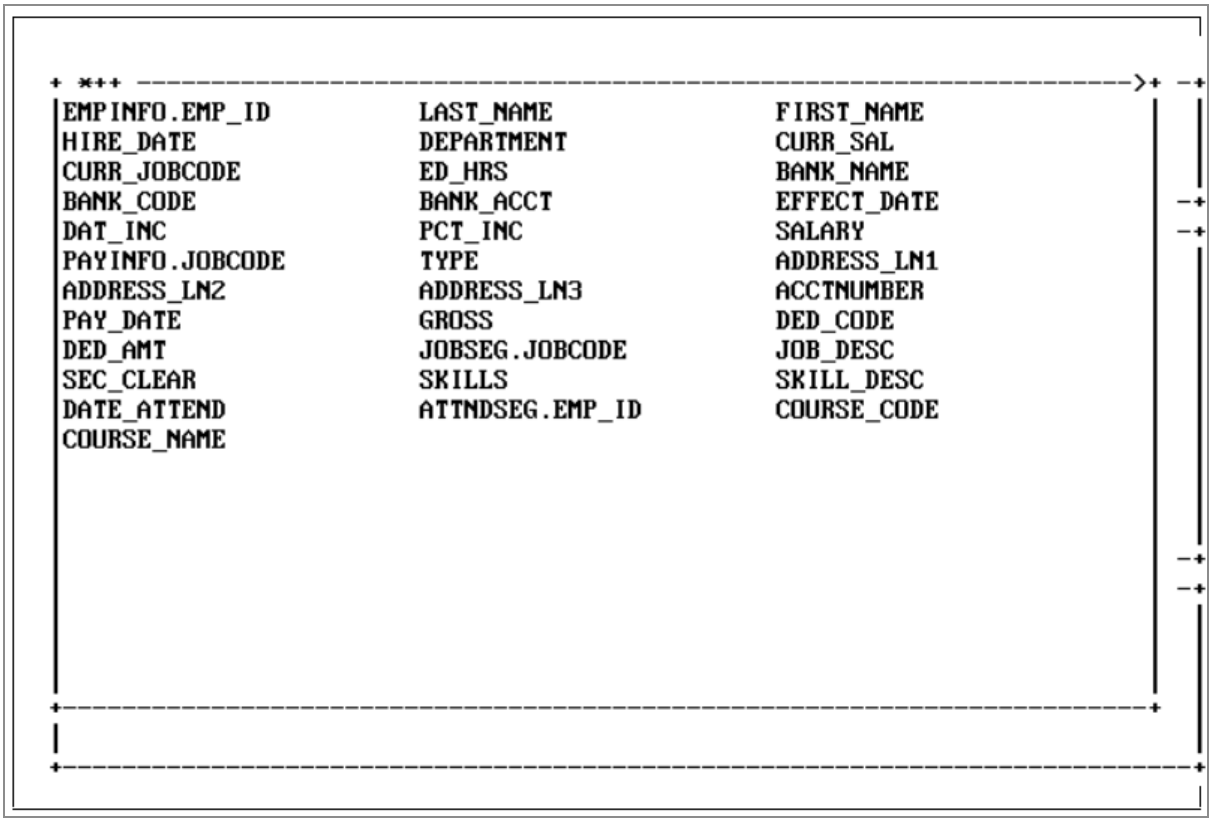
Option	Description
`values of`	
`SUM the COMPUTED values of`	Sums the computed field values that you specify in your report request.

When TableTalk prompts you in this window for the first report column, select:

```
PRINT the individual values of
```

Next, the *Select a data field* window display on the right. If you like, TableTalk enables you to view up to 60 fields on a single screen. You can view this expanded fieldlist by pressing PF4 whenever the *Select a data fiel*d window is active.

The screen displays as shown in the following image:



This is a selection window. Fields may be chosen using PF9 for multi-selections or by placing the cursor at a field and pressing Enter. All of the TableTalk function keys remain available. The PF6 key is a toggle which sorts the fieldlist three ways: in database order, in alphabetic order, and qualified by segment. Symbols in the upper left corner indicate the type of fieldlist sorting: *++ indicates the database order, +*+ indicates an alphabetic order, and ++* indicates segment order. The PF10 and PF11 keys enable you to move left and right within the fieldlist window. (Note that to activate the sort field option, SET FIELDNAME must be set to NEW.) When you press PF4 again, you return to the original *Select a data field* window.

From the *Select a data field* window on the right, select three fields, EMPINFO.EMP_ID, LAST_NAME, and FIRST_NAME. To speed selection, use PF9, the Multi-Select feature, to select the three fields. Do not press Enter until you have finished selecting all fields with PF9. Since the cursor is on

```
EMPINFO.EMP_ID
```

press PF9. The field is displayed in the bottom window. Then, move the cursor to

```
LAST_NAME
```

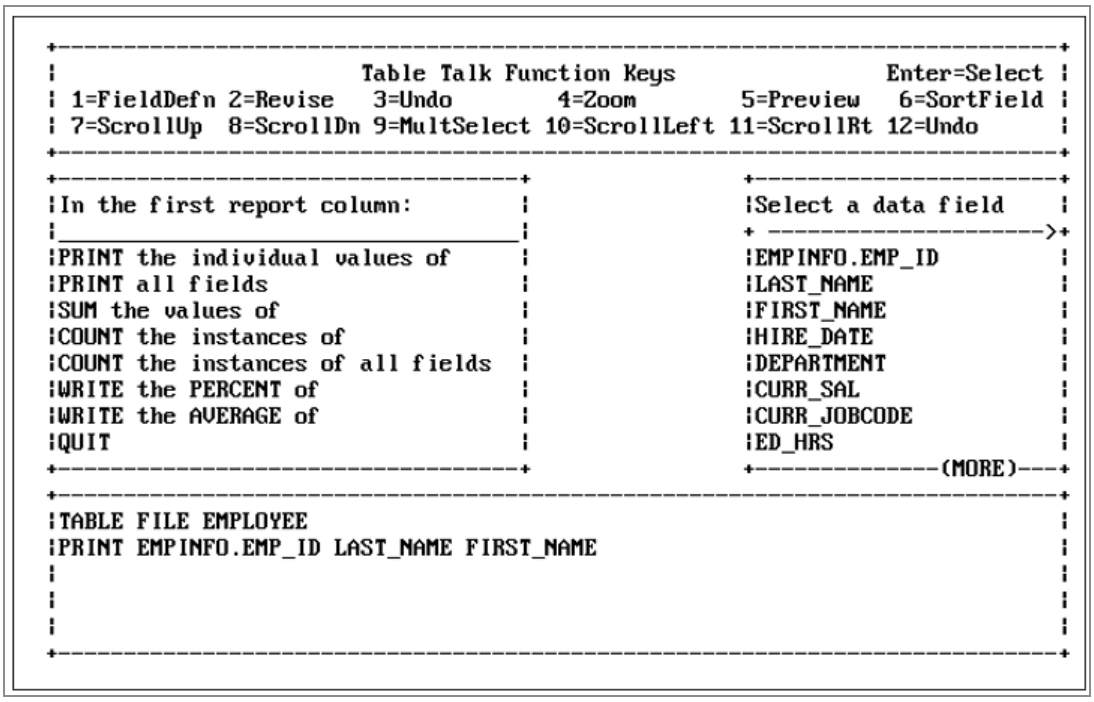
and press PF9.

Finally, repeat the process for:

```
FIRST_NAME
```

Press Enter.

Each field displays in the bottom window after you press PF9. Remember, when you use the Multi-Select facility to select more than one item from a list, you use PF9 to execute your choice. When you select single items from a list, use Enter to transmit your choice.

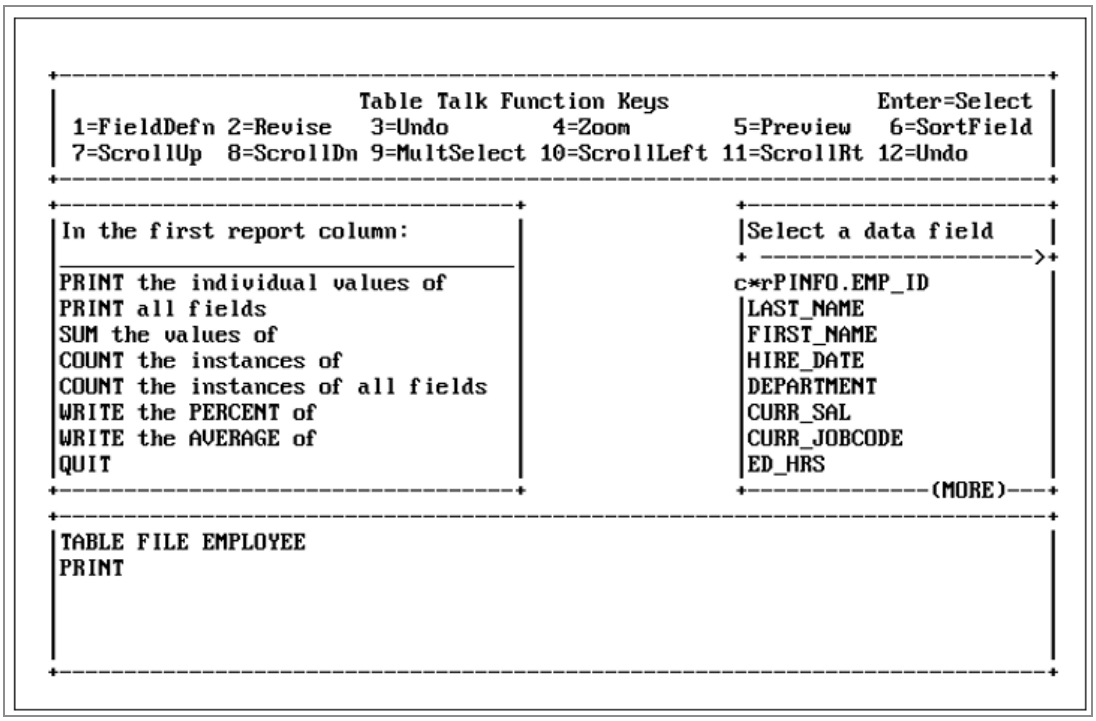


You can also search for an initial or beginning pattern of a fieldname within the fieldlist window or the zoomed fieldlist window. The search pattern, which must always begin on the window border, uses an asterisk (*) or a question mark (?) as the wildcard. An asterisk (*) means that there may be any number of characters in that part of the string. A question mark (?) is a placeholder for a single character. The wildcard character may be placed in any position of the string. An equal sign (=) is used to repeat the last search pattern.

If there are several options that begin with the letter you pressed, the cursor rests at the first item that meets the screening criteria.

If a selection is found on the zoomed fieldlist screen, the cursor rests at the first column of the row where the field is found.

The following image illustrates a search for a string that begins with the letter *C* and ends with the letter *R*:



Note: FOCUS handles fieldnames with a maximum length of 66 characters. The name may include qualifiers for the Master File name and the segment name. For example, the name EMPINFO.EMP_ID identifies the field EMP_ID in the segment EMPINFO.

After completing your selections and pressing Enter, the following options are now available to you in the *And in the next report column* window. You will notice that the options in this window vary, depending on what you select from the preceding screen.

Option	Description
`No more`	Progresses to the next window in the session.
`And PRINT the individual values of`	Redisplays the *Select a data field* window for selection of additional fields.
`And PRINT`	Prints all fields in a single path file and all fields in the left path of a multi-path file.

Option	Description
`all fields`	
`And SUM the values of`	Displays the sum of the numeric fields that you specify.
`And COUNT the instances of`	Displays the count of occurrences of the numeric fields that you specify.
`And COUNT the instances of all fields`	Counts the occurrences of all fields in a single path file and all fields in the left path of a multi-path file.
`And WRITE the PERCENT of`	Calculates the percent for a numeric field you specify and writes the numbers on your report. This option generates a combination of AND and PERCENT syntax in your request.
`And COMPUTE the RATIO of`	Computes the ratio of two numeric fields you specify. This option generates the AND COMPUTE RATIO syntax in your request.
`And COMPUTE the PRODUCT of`	Calculates the product of two numeric fields you specify. This option generates the AND COMPUTE PRODUCT syntax in your request.
`And COMPUTE the DIFFERENCE between`	Determines the difference between two numeric fields you specify. This option generates the AND COMPUTE DIFFERENCE syntax in your request.

Option	Description
`And COMPUTE the SUM of`	Determines the sum of two numeric fields that you specify. This option generates the AND COMPUTE SUM syntax in your request.
`Underneath the last column`	Positions a data field directly underneath another data field on a report. This option generates OVER syntax in your request.
`But first title the last column AS`	Specifies a new report column title for the last column specified. This option generates AS syntax in your request.
`But first format the last column as`	Allows you to change the print format of the last chosen PRINTed or SUMmed field.
`And COMPUTE the general expression`	Creates a numeric operation between two or more numeric entities you specify. This option generates the AND COMPUTE syntax in your request.
`Skip to the END`	Displays the *Do you want to* window that enables you to execute, save, or clear the current request. This option generates no syntax in your request.

Note: If you make a mistake, press PF3 to remove your last selection and return you to the previous TableTalk prompt window.

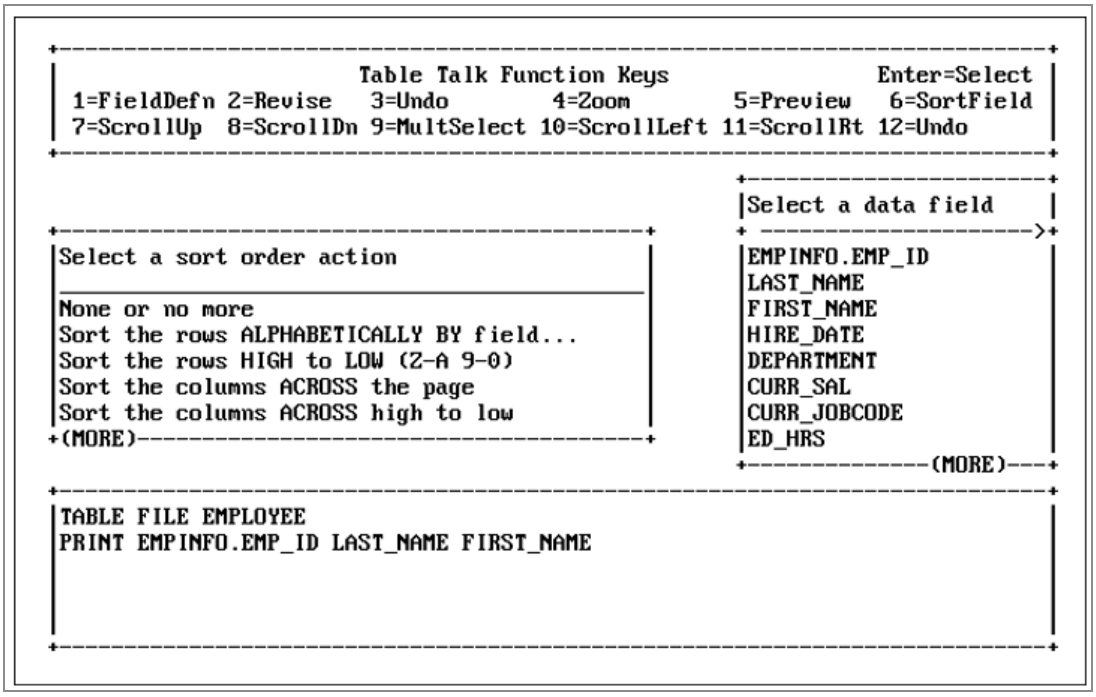
Now, with all three report columns specified (EMPINFO.EMP_ID, LAST_NAME, and FIRST_NAME), select the following option and press Enter:

```
No more
```

The next step is to tell TableTalk how the information should be organized. The following topic describes how you would define a sort field in TableTalk.

Selecting Sort Fields

The windows that follow enable you to select sort fields for your report. The Multi-Select facility (PF9) is not available for selecting sort fields. The first window, *Select a sort order action*, is shown in the following image:



Usually, when you use an alphabetic sort, it will also sort numbers in low to high order. For this request, EMPLOYEE records are sorted alphabetically by DEPARTMENT.

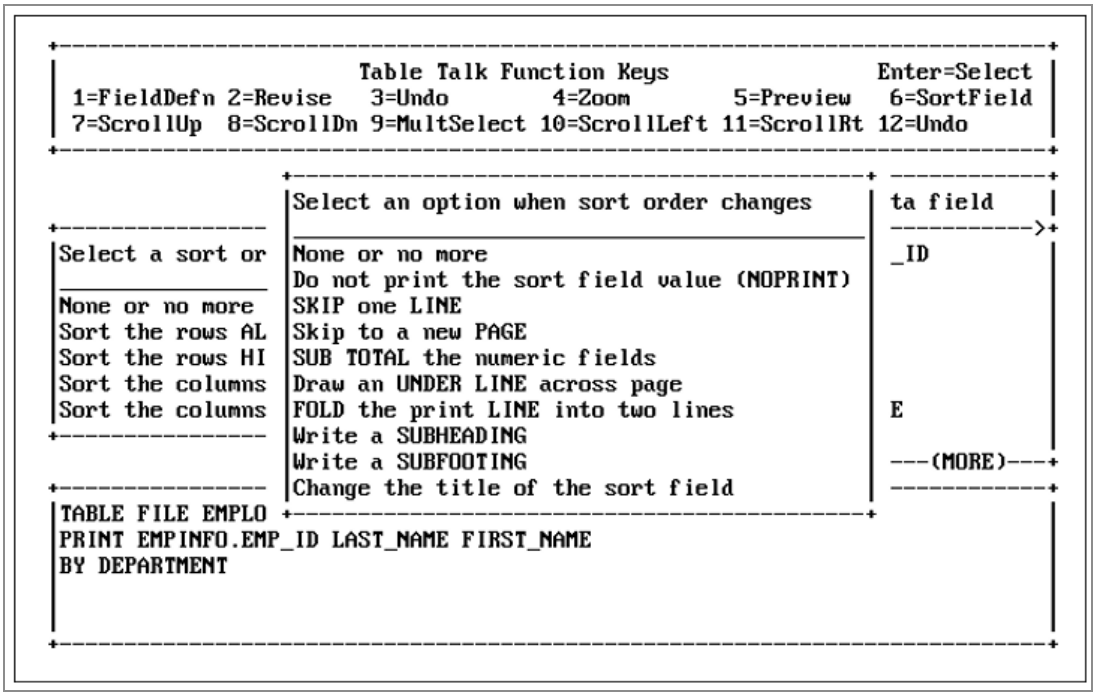
When TableTalk prompts you for a sort order action, select the following option, and press Enter:

```
Sort the rows ALPHABETICALLY BY field...
```

Then, as the sort field, select the following field, and press Enter:

```
DEPARTMENT
```

This brings up the following screen, the *Select an option when sort order changes* window, which displays a list of formatting options available when the sort order changes:



As with any list of options, to select one, you would simply position the cursor next to the desired item and press Enter. Since these options are not applicable to the request you are creating, to return to the *Select a sort order action* window, select:

```
None or no more
```

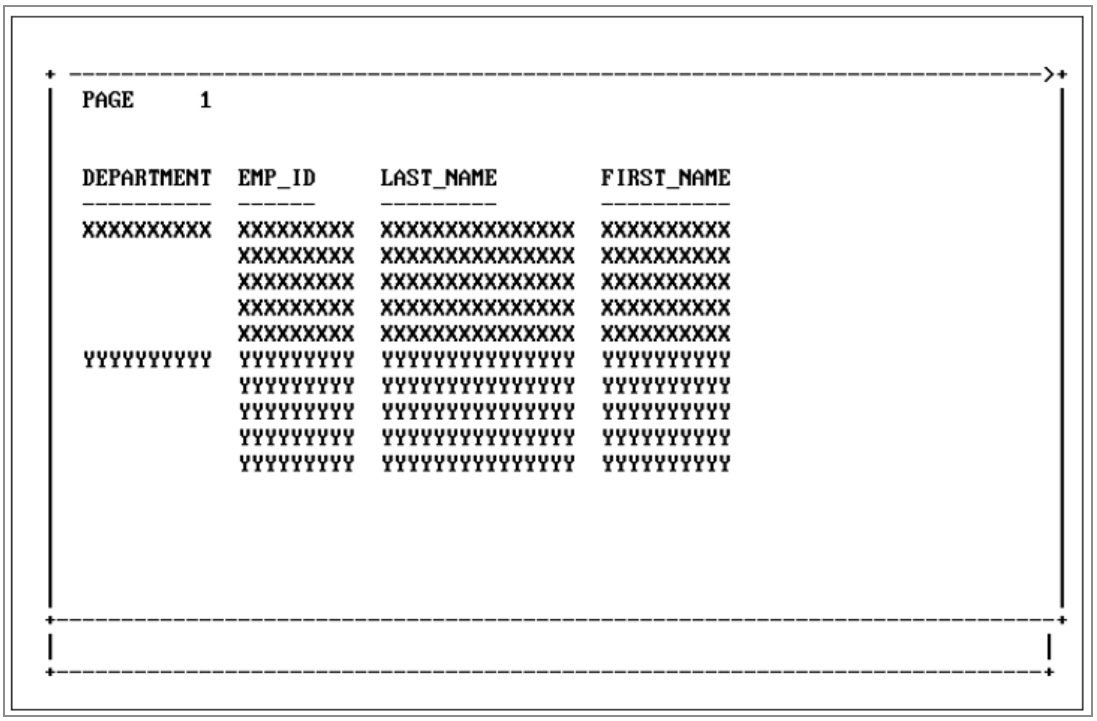
Select

```
None or no more
```

again and press Enter.

At this point, you may be curious as to how the report is starting to look. You can preview a report's layout by pressing PF5 at any time during the TableTalk session. You do not need to finish the report with an END.

This following image shows an example of a previewed request:

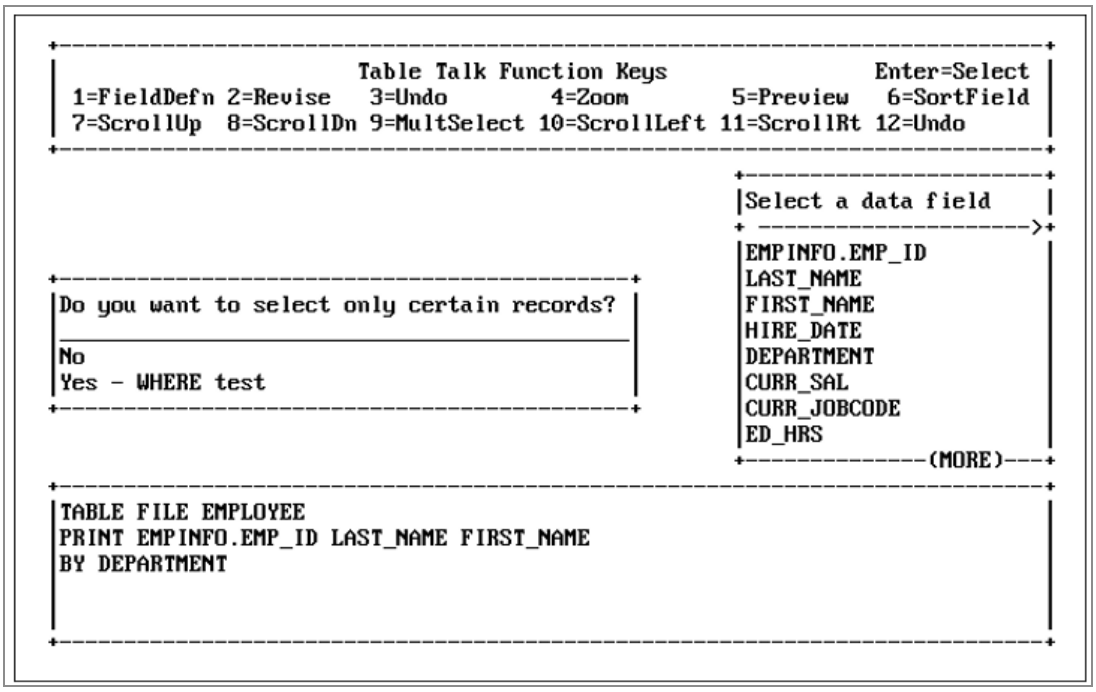


After you have previewed the report, press PF5 again or Enter to return to the *Select a sort order action* window.

Now, with the DEPARTMENT sort field specified, the next step is to tell TableTalk how to test for certain data values. The next topic describes how you would create a record selection test in TableTalk.

Adding Record Selection Tests

The windows that follow enable you to create record selection tests for your report. These tests are built with the FOCUS WHERE keyword, a relation (such as equal to, not equal to, and greater than), and a value. The window that starts the record selection process is shown in the following image.



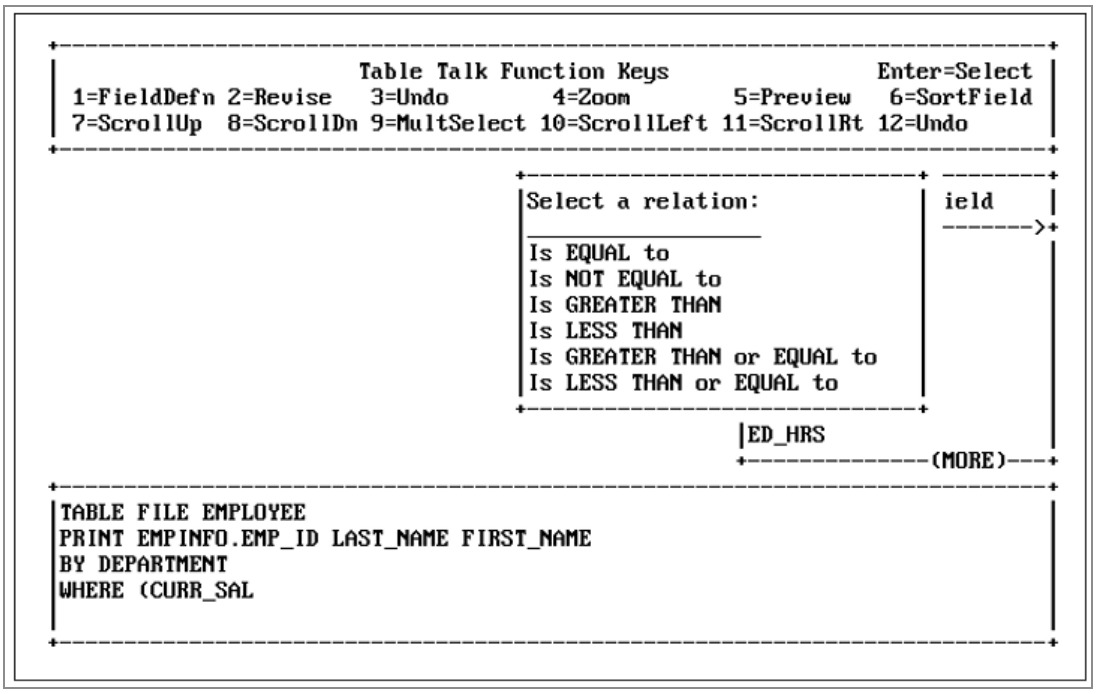
From the *Do you want to select only certain records?* window on the left, select the following option:

```
Yes - WHERE test
```

Then, from the *Select a data field* window on the right, select the following field, and press Enter:

```
CURR_SAL
```

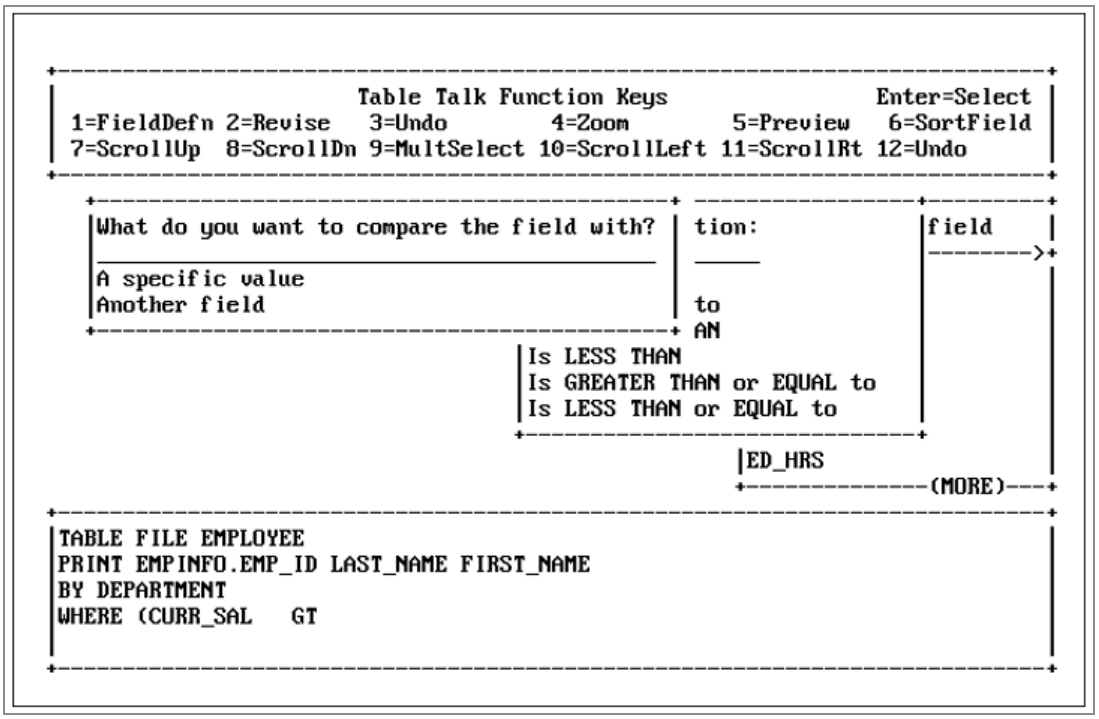
The following *Select a relation* screen prompts you for a relation, as shown in the following image:



As previously mentioned, this request requires records for employees whose annual salary is greater than $25,000. Therefore, select the following relation and press Enter:

```
Is GREATER THAN
```

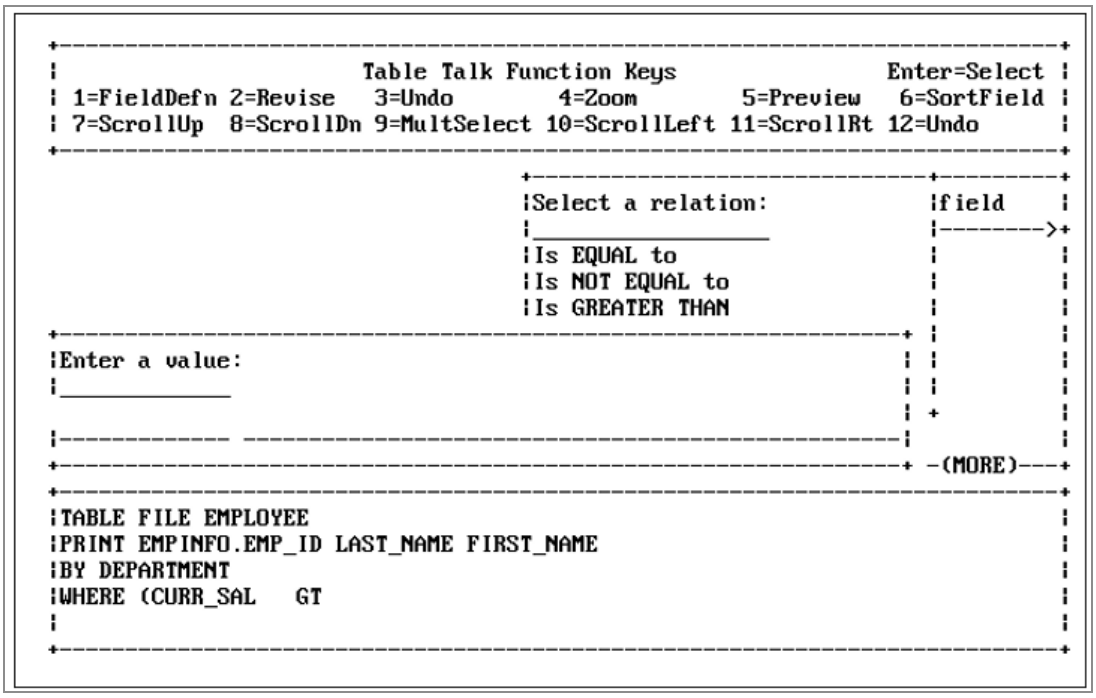
The following window appears and prompts *What do you want to compare the field with?*, as shown in the following image:



If you select a specific value for the test and the current Master File has an ACCEPT attribute, the list is displayed. If the field is numeric, a range is displayed, and if the field is alphanumeric (A), the list of values is displayed. Select

```
A specific value
```

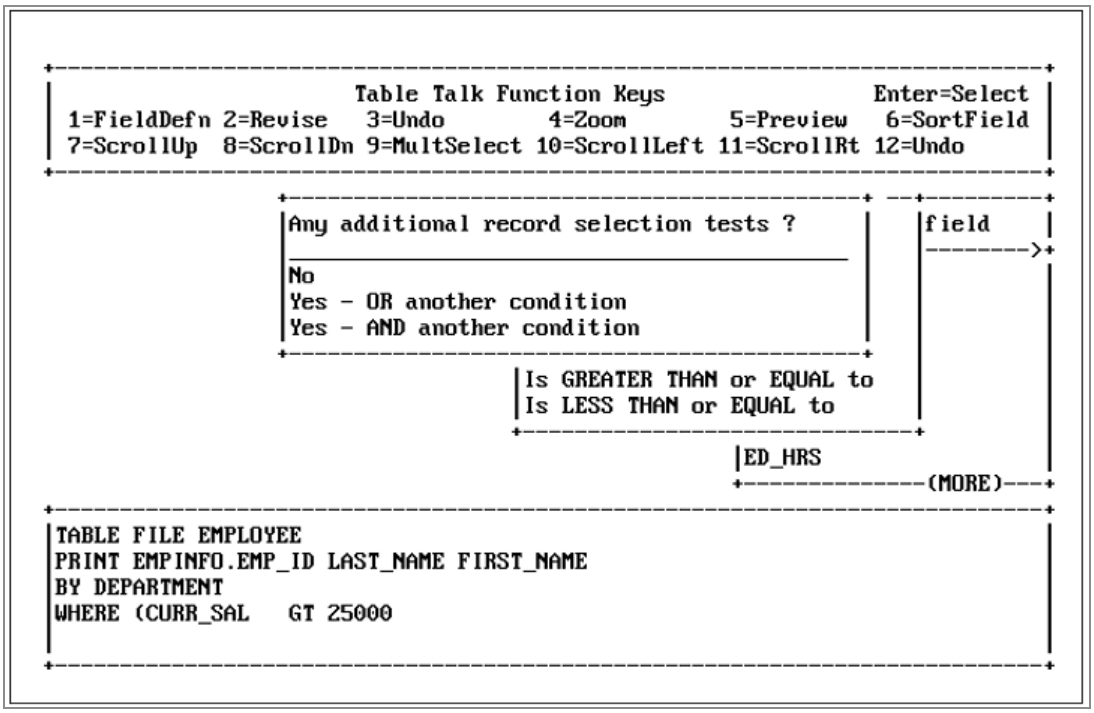
Another window appears prompting you to Enter a value:, as shown in the following image:



Enter the value:

```
25000
```

TableTalk prompts for *Any additional record selection tests?*, as shown in the following image:



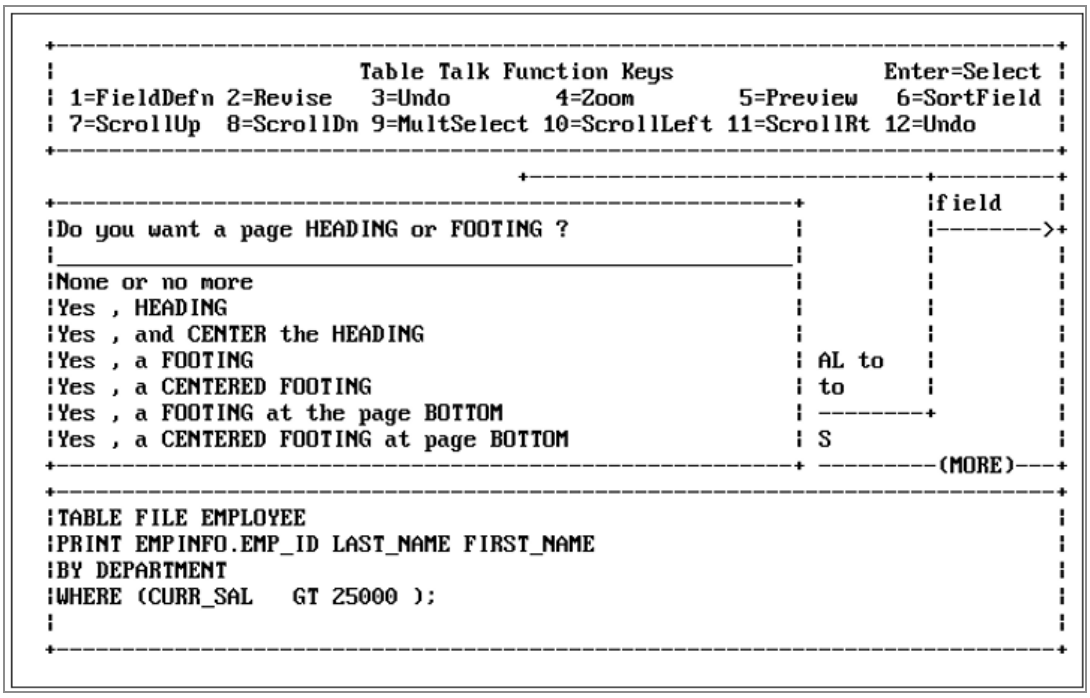
Since this request does not require any additional selection tests, select the following option and press Enter:

```
No
```

Select the *No* option once more, until you see the screen that lets you select headings and footings. With the record selection test specified, the next step is to add optional headings and footings to your report.

Adding Headings and Footings

The windows that follow enable you to create customized headings and footings for your report. The first window that appears is the *Do you want a page HEADING or FOOTING?*, as shown in the following image:



Select the following option:

```
Yes, HEADING
```

TableTalk prompts you to *Enter your text:*. Type the following and press Enter:

```
EMPLOYEES BY DEPARTMENT
```

TableTalk then prompts you for *Any more text lines?*. Choose the following option:

```
No
```

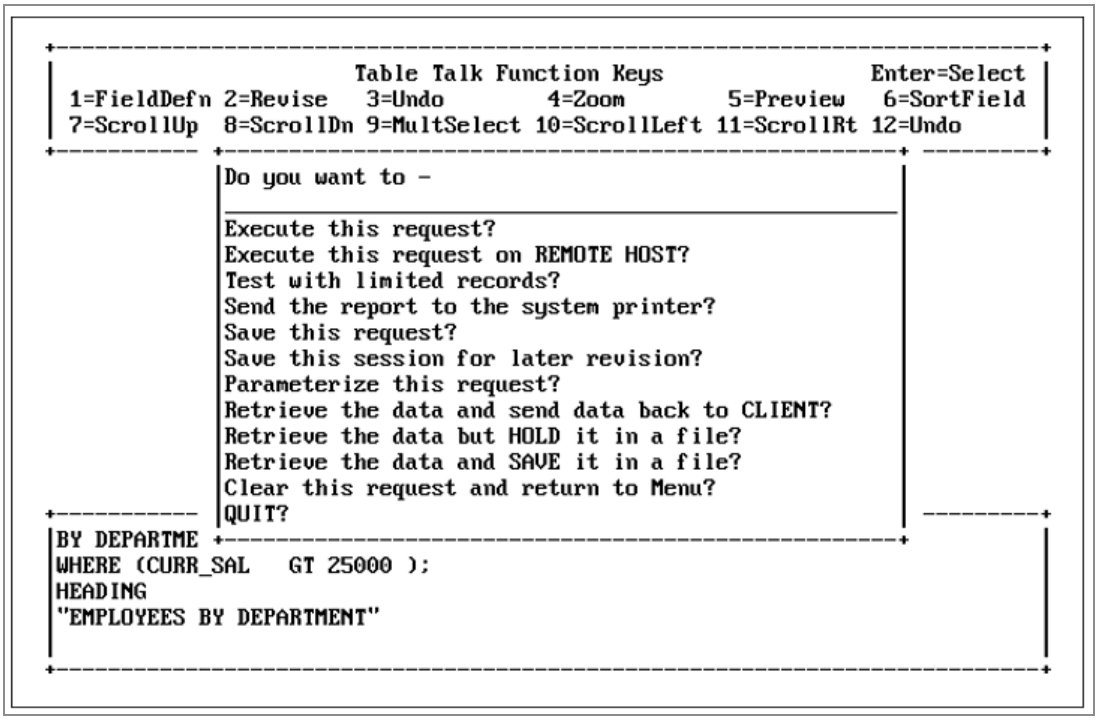
Then, tell TableTalk that there are no additional headings and footings in this report by selecting the following option and pressing Enter:

```
None or no more
```

Now, with the report heading specified, the next section describes the last step in creating a report request.

Executing or Cataloging Your Request

The following screen displays the last TableTalk window and its options. The *Do you want to* window appears, as shown in the following image:



Each of these options is explained in the following table:

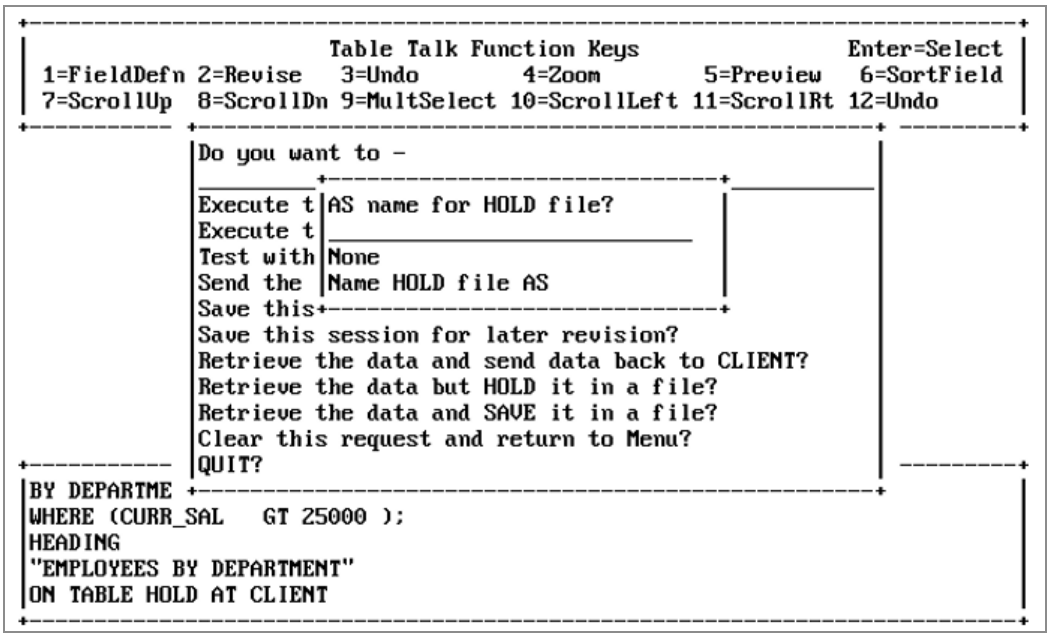
Option	Description
`Execute this request?`	Executes the report request immediately, displaying first the number of data records retrieved and then the report itself on the terminal screen.
`Execute this request on REMOTE HOST?`	Use when Mainframe FOCUS is a client to an **EDA server**. It causes the request to be executed on the server. You will receive an error message if **EDA** is not installed.
`Test with limited`	Enables you to specify, in a subsequent window, the maximum number of records to be retrieved in a report. If you choose this option, TableTalk temporarily incorporates the following FOCUS

Option	Description
`records?`	syntax into your request, and prompts you to *Enter RETRIEVED RECORD LIMIT*, which is the maximum number of records you want to retrieve. `IF RECORDLIMIT EQ` The RECORDLIMIT specification is retained in a saved TABLE REQUEST generated by TableTalk.
`Send the report to the system printer?`	Sends the report to the attached system printer.
`Save this request?`	Stores the report request in a file that you name in a subsequent window. You can execute this saved report later from the FOCUS command level by typing the following command: `EX filename` Under MVS, you may select from a list of existing partitioned datasets allocated to the ddname FOCEXEC, to which you may write this file as a member. If there is no such dataset, that is, a dataset conforming to FOCUS naming conventions where DISP=OLD or NEW, the file is written as a sequential dataset. Any dataset on the displayed list marked by an asterisk (*) was automatically allocated by FOCUS and is not allocated to ddname FOCEXEC. If you specify a pre-existing membername in a partitioned dataset, you will be prompted to supply a new name or overwrite the existing member. Existing sequential datasets will simply be overwritten.
`Save this session for`	The TableTalk session can be saved and edited later using the TableTalk EDIT command.

Option	Description
`later revision?`	Under MVS, the command file is written as a member in the PDS allocated to the ddname FOCEXEC if it is allocated OLD or NEW. If FOCEXEC is allocated as SHR, or is not allocated, the command file is written to the dataset *prefix.FOCEXEC.DATA*, where *prefix* is your high-level qualifier. If this dataset does not exist, commands are written to a sequential dataset. The TableTalk session is written as a member in the PDS allocated to the ddname TTEDIT if it is allocated OLD or NEW. If TTEDIT is allocated as SHR, or is not allocated, the session file is written to the dataset “prefix.TTEDIT.DATA”. If this dataset does not exist, the session is written to a sequential dataset. However, the session will not be available for editing. To save and edit TableTalk sessions, allocate a PDS to the ddname TTEDIT as OLD. An allocation for TTEDIT is not needed if the PDS *prefix.TTEDIT.DATA* exists. The PDS should have LRECL=80, BLKSIZE=1760, RECFM=FB. To edit saved TableTalk sessions under MVS, enter the following command and specify the filename of the session you want to edit: `TABLETALK EDIT filename` If you omit the filename, a list of all TTEDIT files is displayed. The session you want to edit is placed in Revise Mode.
`Parameterize this request?`	This option lets you create an amper variable to allow for user input when the report is run. You can enter up to 70 characters of text to prompt the user. When this option is chosen, you are placed in Revise mode. Parameterization is available as *Parameterize* on the Actions menu of Revise mode. In Revise mode, place the cursor on the value or field to be parameterized and press Enter. You may also use PF7 and PF8 to move around the screen. Then select *Parameterize* from the Actions menu and press Enter. Enter the variable name you wish to use. Next, enter the text you wish displayed for the amper variable and press Enter.

Option	Description
`Retrieve the data and send data back to the CLIENT?`	Adds the line ON TABLE HOLD AT CLIENT to a request intended for use when Mainframe FOCUS is a client to an **EDA server** and prompts you to name the temporary file. If you choose *None*, the default name for the temporary file is HOLD. To specify a name other than HOLD, choose *Name HOLD file as* and type a name (with a maximum of eight characters) on the subsequent screen. After you choose whether or not to name the file, you are prompted to select a HOLD file format.

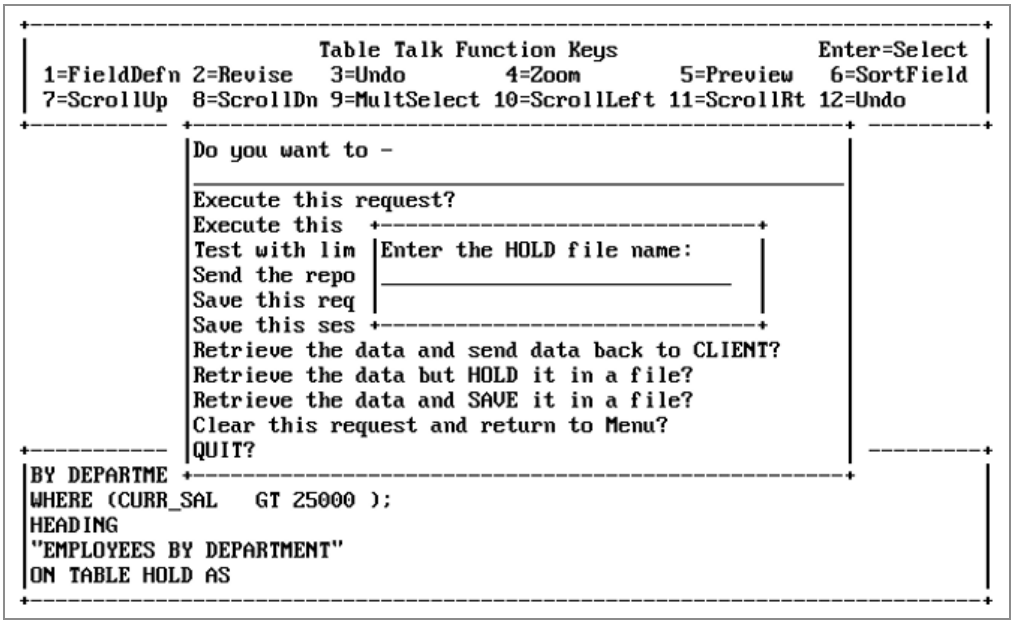
After choosing a format for the HOLD file, you usually choose an option to execute on a remote host, as shown in the following image. This causes the request to run on the server and bring the data back to the Mainframe.



`Retrieve the data but HOLD it in a file?`	Prepares the report and puts it in a temporary file named HOLD, from which other requests can be directed. The HOLD file remains usable for the duration of the FOCUS session, and will appear on the list of available Master Files the next time you enter TableTalk.

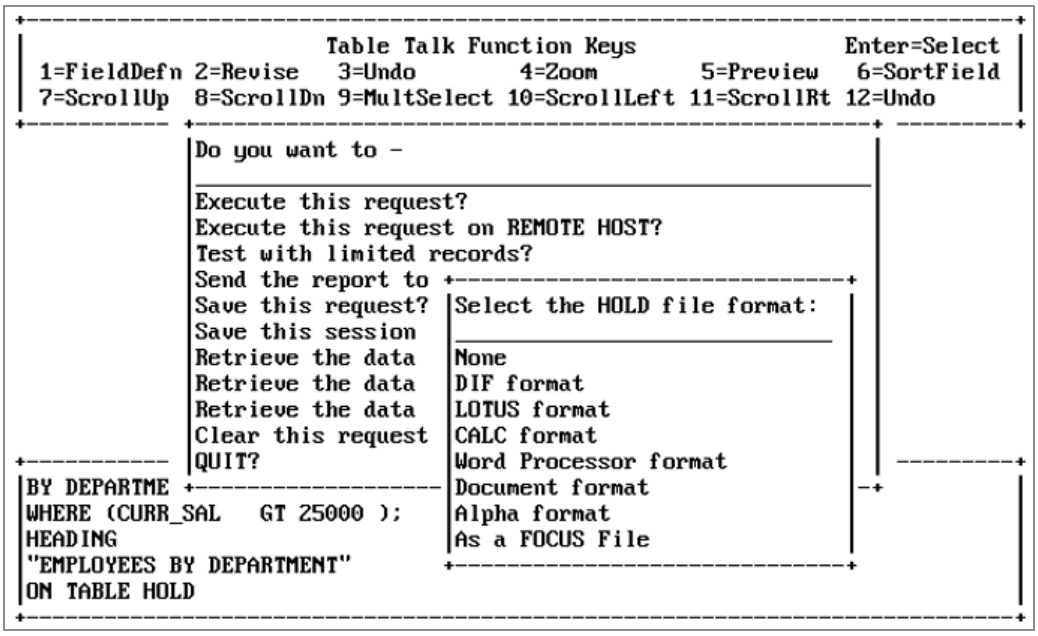
Option	Description

To specify a name other than HOLD, choose *Name HOLD file as* and type a name (with a maximum of eight characters) on the subsequent screen, as shown in the following image.



You can select the HOLD file format from a subsequent window that displays the following choices, as shown in the following image:

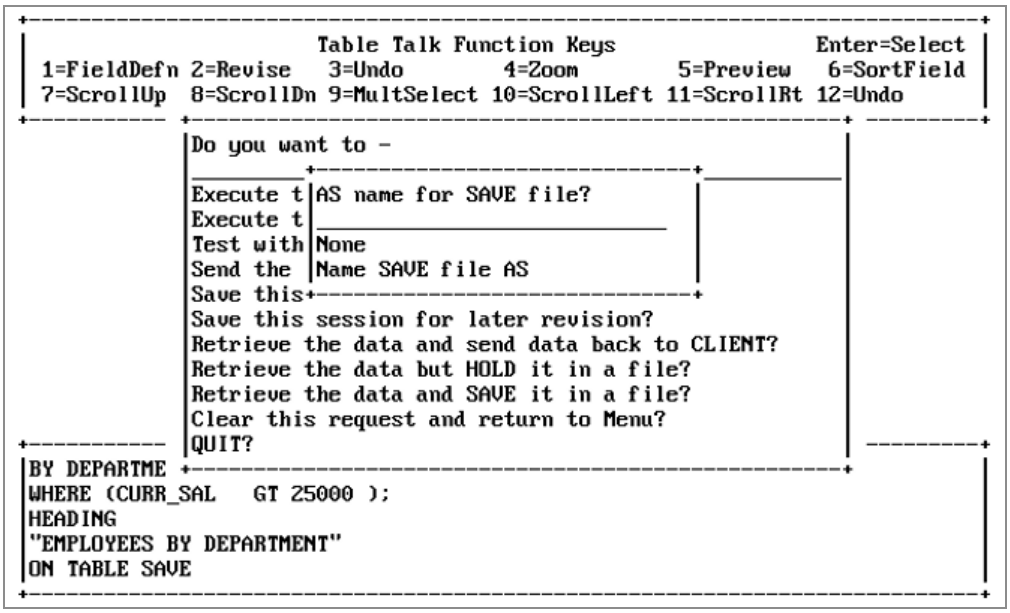
Option	Description

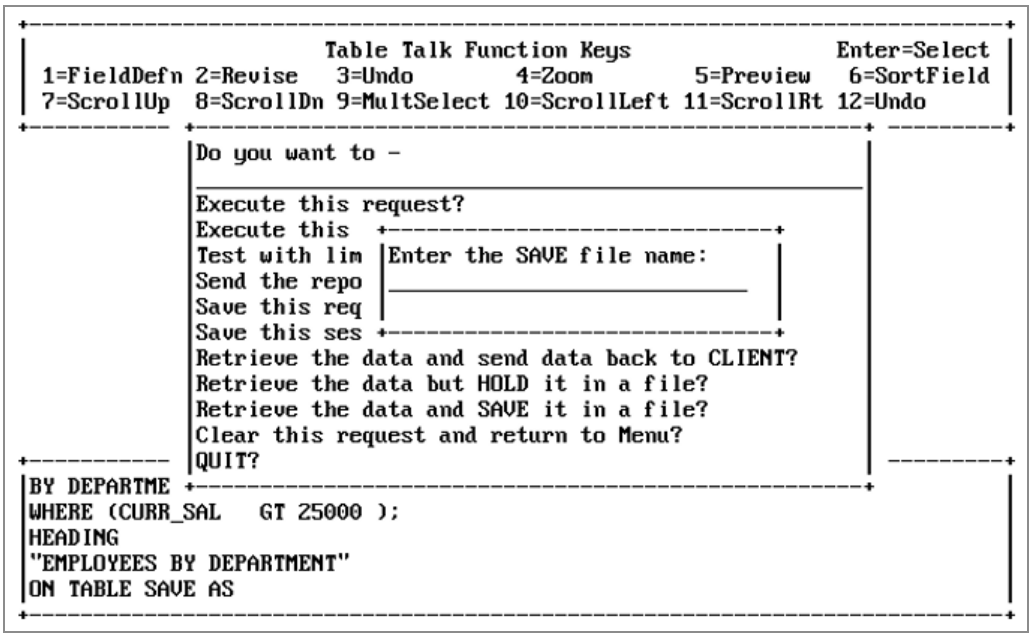


`Retrieve the data and SAVE it in a file?`	Under MVS, FOCUS dynamically allocates a temporary sequential dataset under ddname SAVE. FOCUS allocates five tracks each for primary and secondary space. Record format is variable blocked with record length and block size dependent on the record size. Once DCB attributes are assigned, they remain in effect for the duration of the session even if another SAVE is done to the same file. To keep the file, use the TSO COPY command.

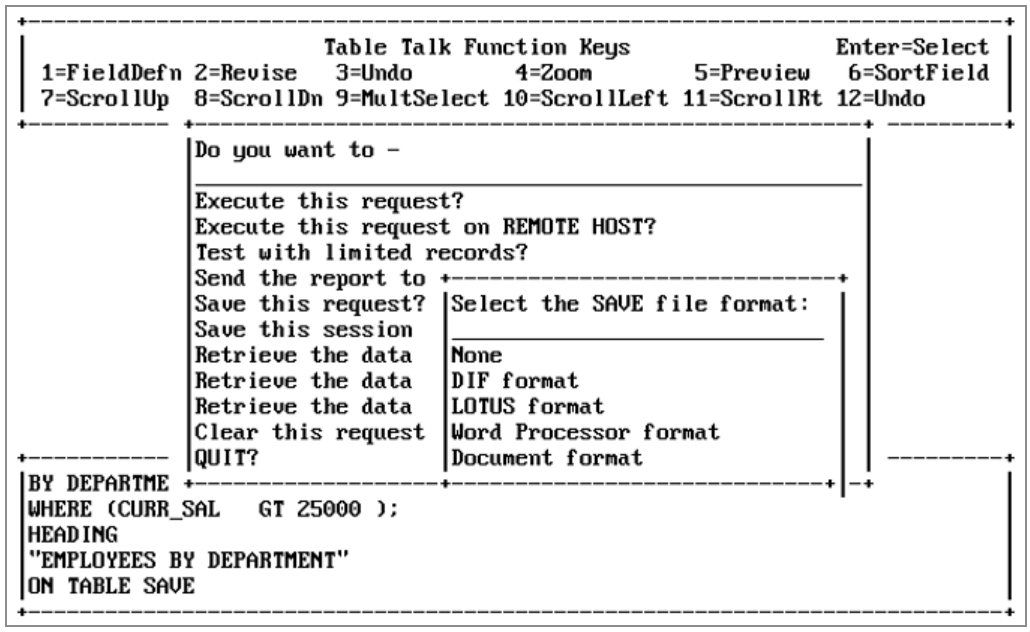
To specify a name other than SAVE, choose *Name SAVE file as* and type a name (with a maximum of eight characters) on the subsequent screen, as shown in the following images.

Option	Description





You can select the SAVE file format from a subsequent window that displays the following choices, as shown on the following image:

Option	Description
	
`Clear this request and return to Menu?`	Erases the current request and starts a new TableTalk session. To avoid losing your work, save the request before clearing it. You can generate a new request using the same file or a different one.
`QUIT`	Ends your TableTalk session without saving the report or its request.

Executing the Request

Select the following option and press Enter:

```
Execute this request?
```

TableTalk displays the number of records you have selected and the number of lines in the report, as shown in the following image.

```
NUMBER OF RECORDS IN TABLE=        3  LINES=      3

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

Press Enter again to display the following output, as shown in the following image:

```
PAGE     1

EMPLOYEES BY DEPARTMENT
DEPARTMENT  EMP_ID     LAST_NAME        FIRST_NAME
----------  ------     ---------        ----------
MIS         818692173  CROSS            BARBARA
PRODUCTION  119329144  BANNING          JOHN
            123764317  IRVING           JOAN









                                END OF REPORT
```

Note that this report displays all the report columns, sort fields, and record selection data you have specified in the previous sections.

When you have finished viewing the report, press Enter to resume your TableTalk session from the last window.

If the request has been saved, you can also execute your request outside of TableTalk by typing the following command at the FOCUS command prompt and pressing Enter. See Executing or Cataloging Your Request for further details.

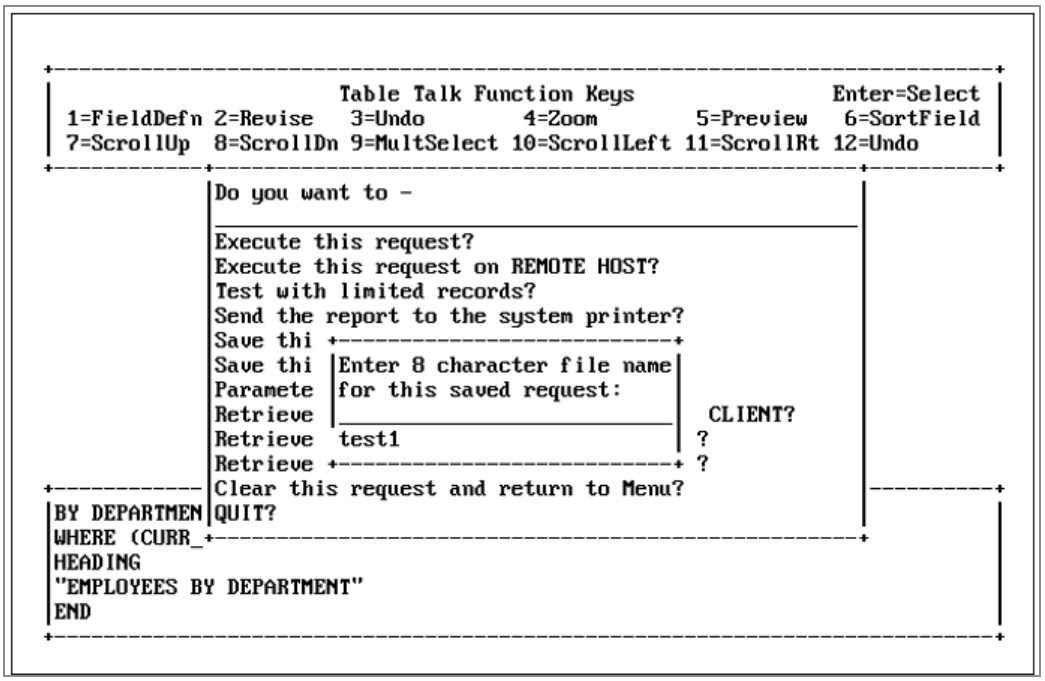
```
EX focexecname
```

Saving the Request

Now that you have executed your request and read the report, you may want to save it for future use. Select the following option and press Enter:

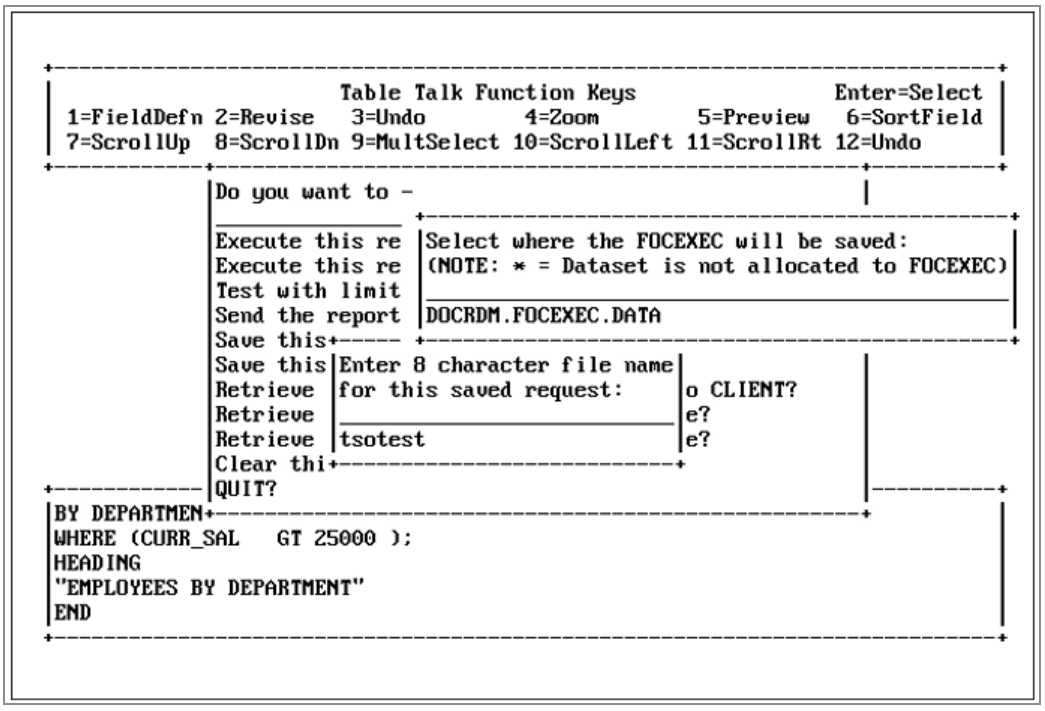
```
Save this request?
```

Type in the name you want for the file and press Enter, as shown in the following image.



- Under MVS, the file is saved to the member of the partitioned dataset allocated to the ddname FOCEXEC.

When you save a file, TableTalk displays the following additional window in which you select the partitioned dataset, as shown in the following image:



Revising Your Report Request

You can use the following to revise a request:

- TableTalk Revise Mode
- TED
- System editors

TableTalk Revise Mode is intended for simple changes to a report request. If you know that you will edit your report request at a later time using Revise Mode, select the following option from the last window:

```
Save this session for later revision?
```

TableTalk will prompt you for a filename, and automatically save both your request and your current TableTalk session.

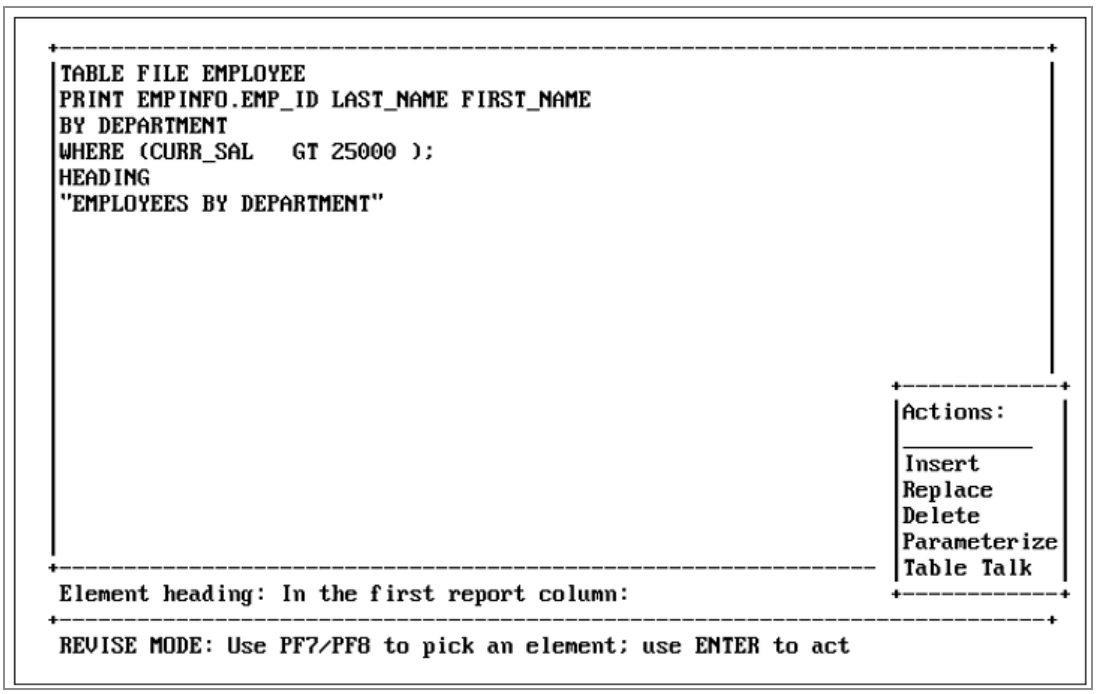
If you issue the following command at the FOCUS command prompt, it will enable you to revise a specific TableTalk session.

```
TABLETALK EDIT [filename]
```

For extensive changes, you should use TED or your system editor.

Using TableTalk Revise Mode

To edit a request in TableTalk, press PF2. This places your request in Revise Mode. For example, if you pressed PF2 while building the tutorial request, the Revise Mode screen would appear as shown in the following image:



This screen consists of:

- A large center window that displays the report request in FOCUS syntax.
- A cursor that selects the word that is affected by the option you choose from the *Actions*: window in the lower right corner. (Use the PF7 and PF8 keys to highlight the request element you want to change and the up and down arrow keys to select the editing action you want to perform on the highlighted text.)
- A bottom window labeled *Element heading* that identifies the highlighted word as an element of your report request.

- An *Actions:* window in the lower right corner of the screen that displays the types of revisions you can make to the highlighted text. You can insert, replace, delete, and parameterize text in your request, or you can return to the current TableTalk session.

To revise your request, highlight the element in the left window you want to change using PF8 to move forward, and using PF7 to move backward. When the element you want to change is highlighted, select an action from the *Actions:* window using the up and down arrow keys or the Tab key, and press Enter.

For example, to replace the sort field in the sample report request, highlight DEPARTMENT using PF8. Since the cursor is on REPLACE in the *Actions:* window, press Enter.

TableTalk displays a list of data fields from which you select a new key field in Revise Mode. TableTalk replaces the old field with the new field you specify. For this example, select FIRST_NAME; DEPARTMENT is instantly replaced. Return now to TableTalk by selecting it in the *Actions:*window, or by pressing PF2 to display the execute window.

Using TED With TableTalk Requests

When you choose to execute a request, TableTalk automatically saves the information.

- Under MVS, TableTalk saves it to a temporary sequential file allocated to ddname TABLTALK.

If you saved the file before executing it, the filename is the one that you specified.

After executing the request and ending your TableTalk session, you can edit this file in FOCUS using TED, or your system editor.

You can also execute your request from within TED by typing the following at the TED command line and pressing Enter:

```
RUN
```

The RUN command automatically saves and executes the FOCEXEC.

Using Help in TableTalk

In TableTalk, you can display user-supplied help information about files and fields. Brief descriptions of files and fields are available if those descriptions have been stored in

separate information (or HELP) files.

The first screen in TableTalk lists filenames and one-line descriptions of those files for which separate HELP files have been created. If help information is available for the file, it is automatically displayed on your screen.

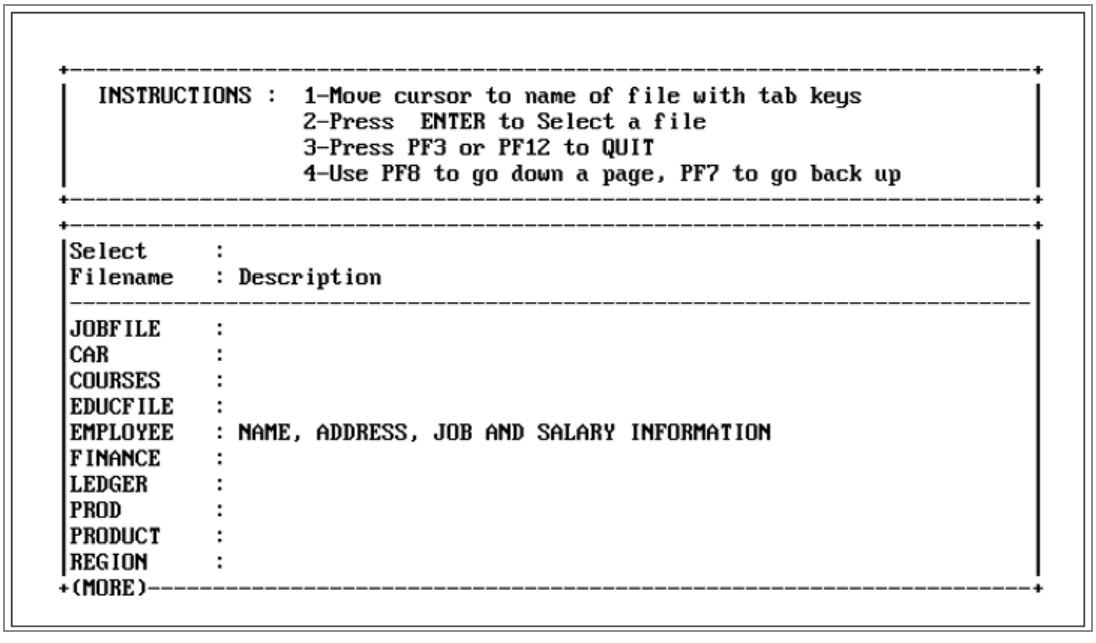
You can also display a fieldname description if a separate HELP file has been created for the file containing the field by moving the cursor to the fieldname and pressing the PF1 key. If you press PF1 again, TableTalk clears the description from your screen.

Creating a HELP File With File Descriptions

To display one-line descriptions of the files listed in the first TableTalk screen:

- Under TSO, FOCFILES is a member in the PDS allocated to the ddname FOCDEF.
- Begin each line of the FOCFILES file with an eight-character filename in uppercase. If the filename is less than eight characters long, you must pad the remaining positions with blanks. Use the remainder of the line for the file description.

The following image shows an example of a file with a file description:



Creating a HELP File With Field Descriptions

To provide information on fields:

- Under MVS, FOCDEF is stored in a PDS with a fixed format that is allocated to the ddname FOCDEF with a member name the same as the description. There is no required LRECL or BLKSIZE.
- Begin each line with a 12-to-66 character field containing the fieldname in uppercase. If the field is less than 12 characters long, you must pad it with blanks.
- You can enter up to three consecutive lines of text for each fieldname. Each line of the description must begin with the fieldname.
- Up to 44 characters of description per line is displayed. More than 44 characters may be included, but only the first 44 characters is displayed.
- Enter the fieldnames in the same order as they appear in the Master File.

The following image shows an example HELP file with field descriptions:

```
EMP_ID       THE EMPLOYEE ID NUMBER
EMP_ID        FOR EVERY EMPLOYEE IN THE FILE
LAST_NAME    THE EMPLOYEE'S LAST NAME
FIRST_NAME   THE EMPLOYEE'S FIRST NAME
HIRE_DATE    THE DATE OF HIRE
DEPARTMENT   THE DEPARTMENT IN WHICH THE EMPLOYEE
DEPARTMENT    CURRENTLY WORKS
CURR_SAL     THE EMPLOYEE'S CURRENT YEARLY SALARY
CURR_JOBCODE THE EMPLOYEE'S JOBCODE
ED_HRS       THE NUMBER OF HOURS THE EMPLOYEE HAS
ED_HRS        SPENT ON EDUCATION
BANK_NAME    THE BANK NAME FOR AUTOMATIC PAYROLL DEPOSIT
BANK_CODE    THE BANK CODE FOR AUTOMATIC PAYROLL DEPOSIT
BANK_ACCT    THE EMPLOYEE'S BANK ACCOUNT NUMBER
BANK_ACCT     FOR PAYROLL PURPOSES
```

Help information can also be created for:

- Temporary fields created with the FOCUS DEFINE command.
- Defined fields from other files that have been joined with the FOCUS JOIN command.

PlotTalk

This topic describes how to create graphs using PlotTalk. PlotTalk is a window-driven utility that guides you through the process of creating graphs using data from your file or view. It enables you to generate graphs without knowing FOCUS GRAPH syntax. Depending on your system, you can create up to five types of graphs:

- Continuous curve graphs
- Vertical bar charts
- Vertical bar charts
- Horizontal bar charts
- Scatter of values diagrams
- Pie charts

You can create graphs from FOCUS or non-FOCUS database files. These graphs can be displayed on your screen or sent to a printer. The requests generated with PlotTalk can be saved in files for future revisions.

This content guides you step-by-step through the creation of a simple GRAPH request. It will produce a continuous curve graph using data from the EMPLOYEE data file, and show the average salaries of all employees earning more than $15,000. You gain hands-on experience on how to select the fields and screening conditions to create such a graph.

PlotTalk Requirements

In order to use PlotTalk, you must have a Master File and the data file it describes. See Introducing Talk Technology for a discussion of these requirements and steps used to create a data file.

Graph requests created with PlotTalk are limited to 24 lines. When you reach the 24-line limit, you can execute or save the request, among other options discussed later in this content. Graph requests created or edited outside of PlotTalk have no line limit.

Talk Technology products convert all input into uppercase text. In PlotTalk, this may affect computations, record selection tests, axis labels, headings, and footings.

Note that your particular operating system and configuration may have different requirements from those described here. If information in this content does not apply to your system or configuration, consult the *FOCUS for IBM Mainframe User's Manual* and/or your FOCUS administrator for further instructions.

If you are a GDDM®, or ICU user, you must:

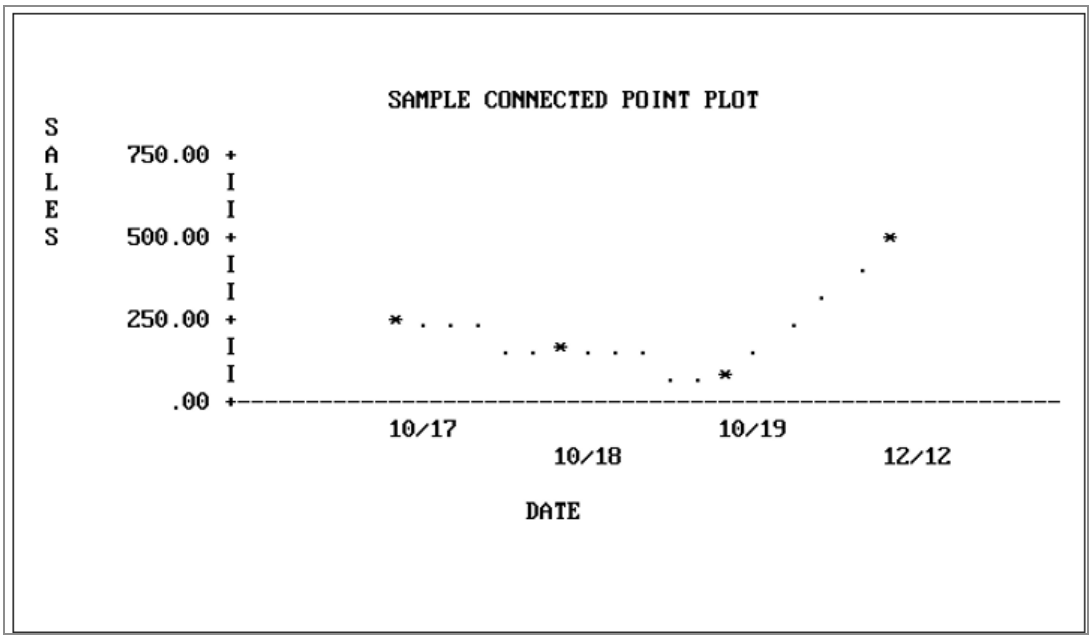
- Have an extended attribute terminal (for example, 3279) with enough PS (Programmed Symbols) storage to run GDDM.
- Have 4M of main memory storage.
- Under MVS, allocate the GDDM runtime library to STEPLIB or USERLIB.
- Under MVS/TSO, allocate PTEDIT and TTEDIT as two partitioned datasets *prefix.TTEDIT.DATA* and *prefix.PTEDIT.DATA* with LRECL=80, BLKSIZE=1760, and RECFM=FB.

Sample Graphs

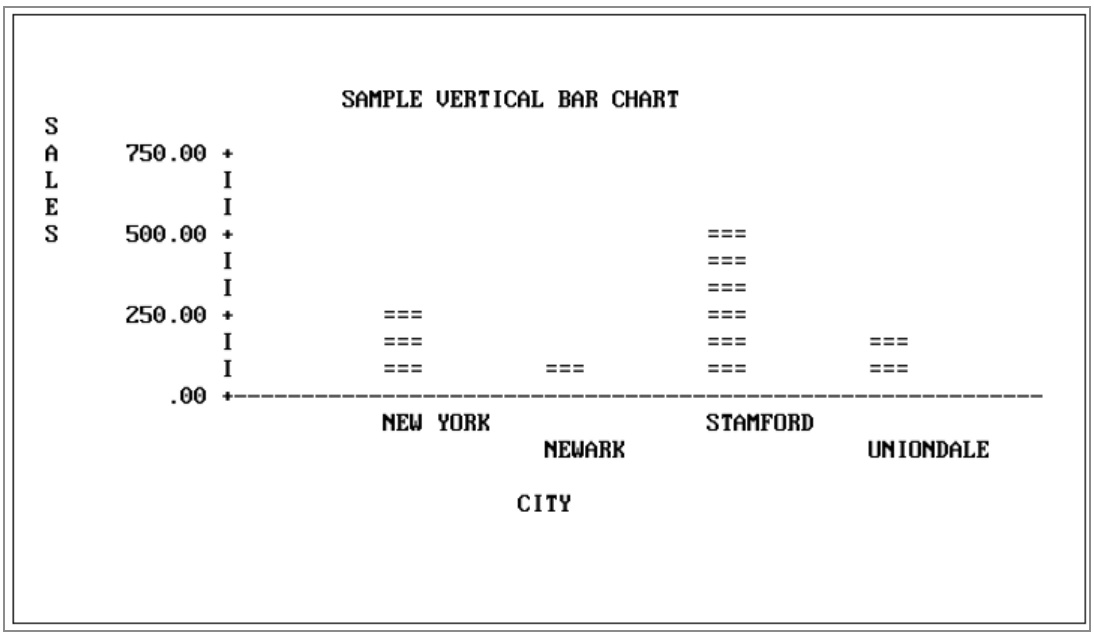
The following screens show the different types of graphs you can create with PlotTalk. Standard terminal graphics were used for all examples except for the pie chart, which was created using **ICU high-resolution graphics, Version 2.0, Release 2.0**. Note: ICU and GDDM versions of the other graph types (continuous curve, vertical bar chart, horizontal bar chart, and scatter of values diagram) differ slightly from the examples shown.

PlotTalk adapts graph output to take advantage of your terminal capabilities if FOCUS was installed to work with **ICU and GDDM**, producing the most advanced output possible without user intervention.

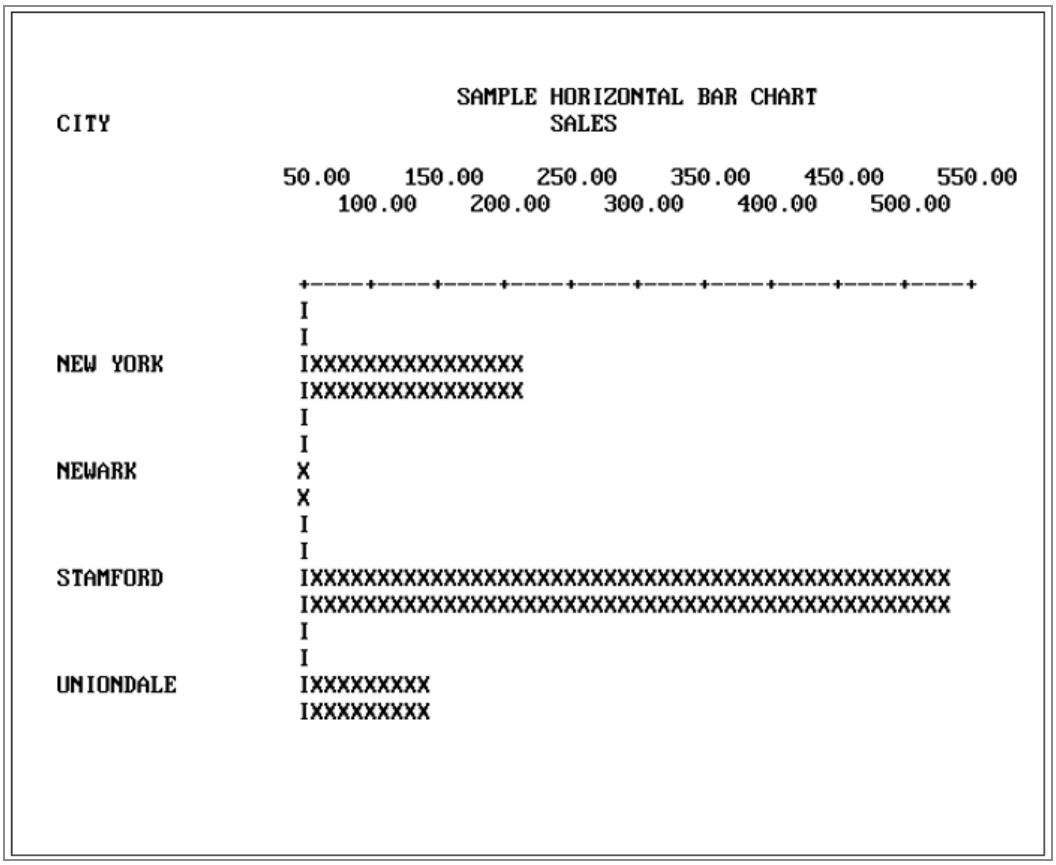
The following image shows a continuous curve graph:



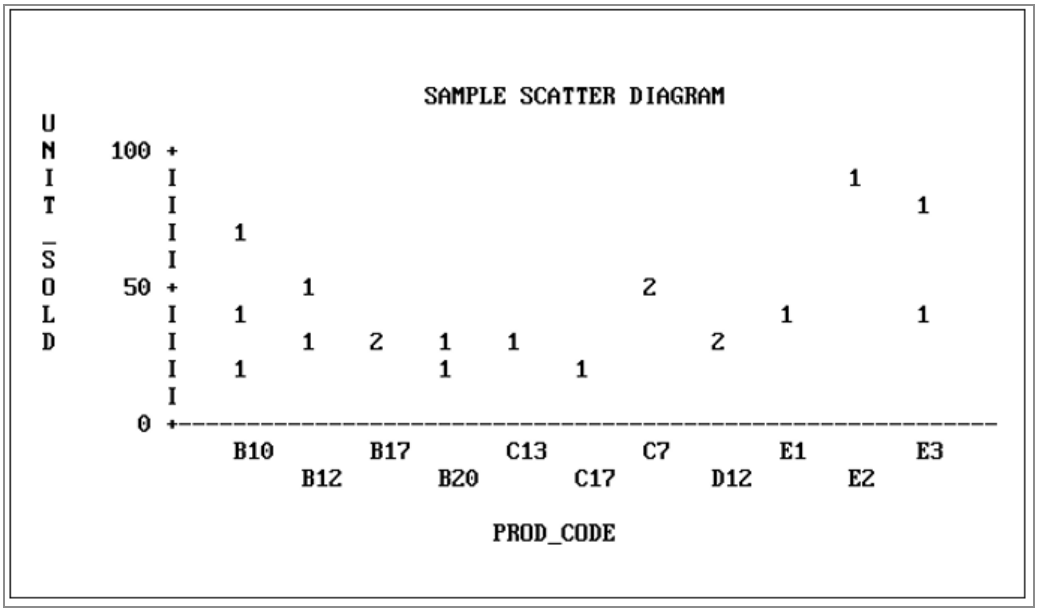
The following image shows a vertical bar graph:



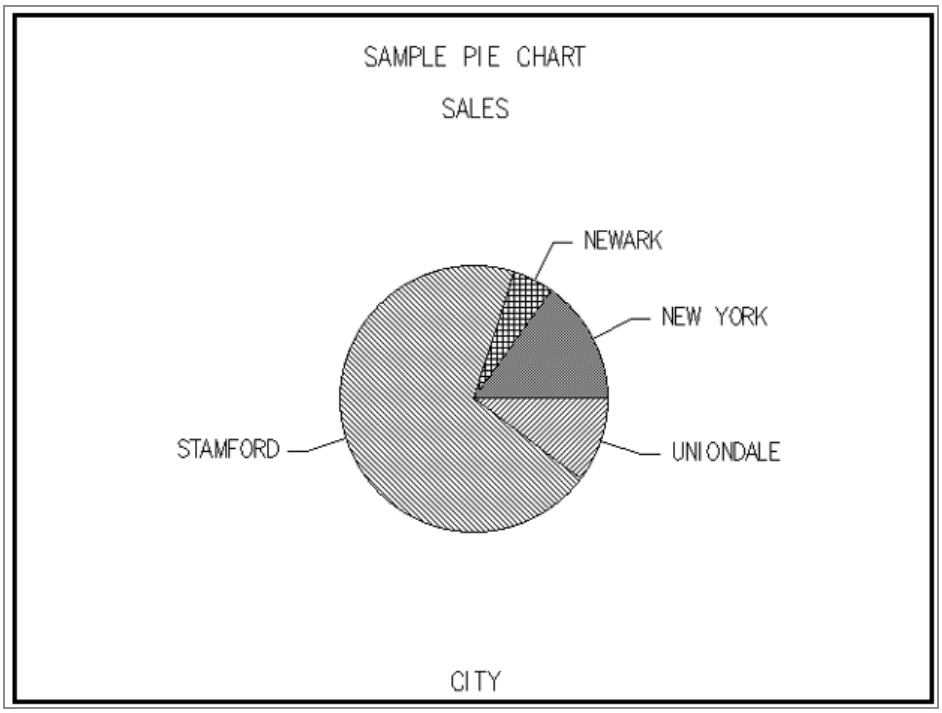
The following image shows a horizontal bar graph:



The following image shows a scatter of values diagram:



The following image shows a pie chart:

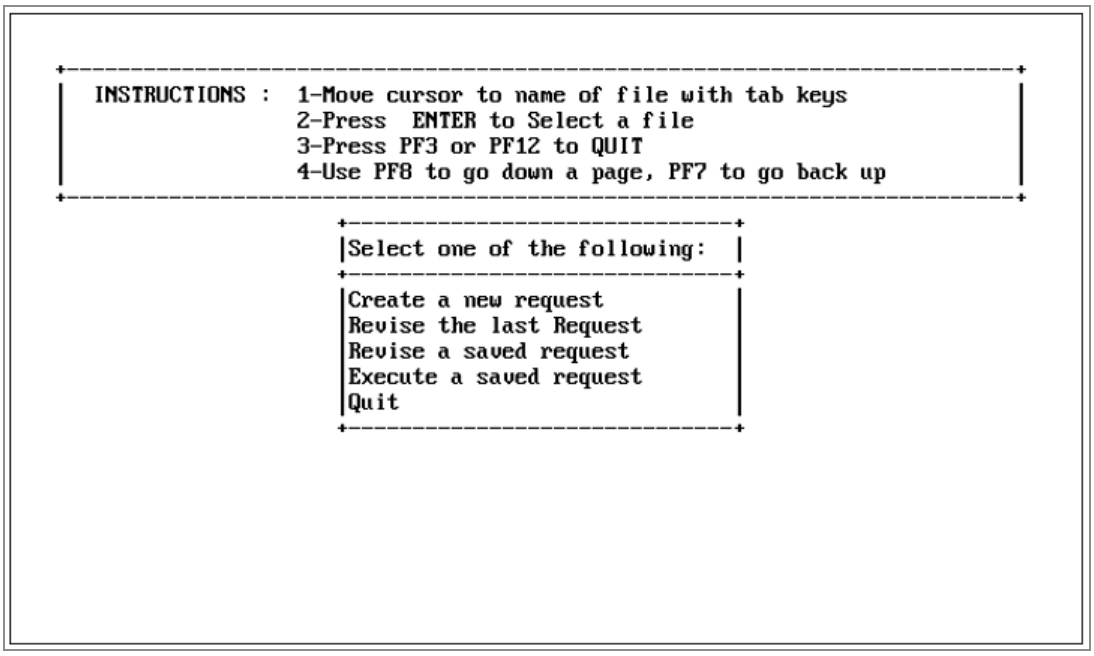


Entering PlotTalk

There are several ways you can invoke PlotTalk. The first way to invoke PlotTalk is to type the following command at the FOCUS command prompt and press Enter.

```
PLOTTALK
```

The first screen that you see is the PlotTalk *Select one of the following:* window, as shown in the following image:

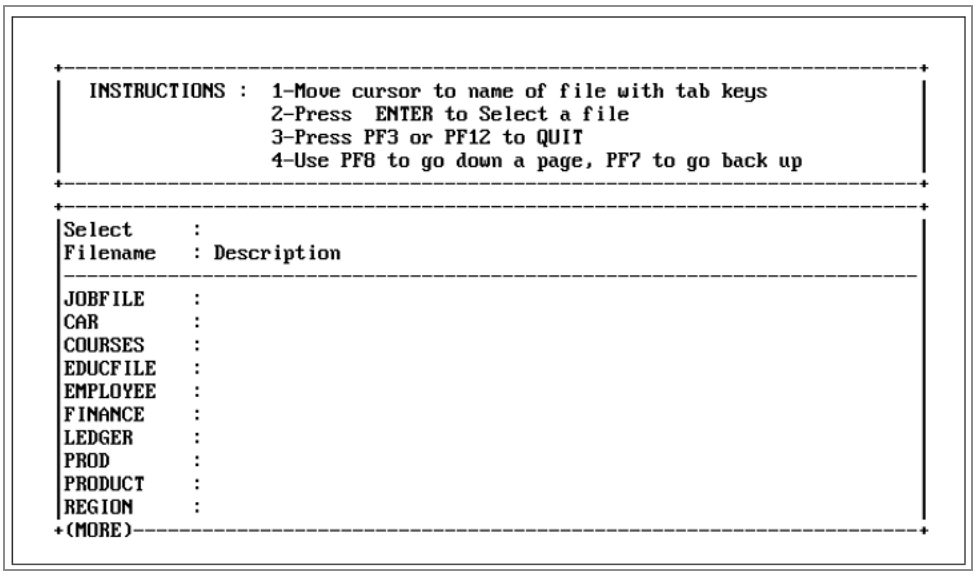


Select the option

```
Create a new request
```

The next screen that you see is the PlotTalk *File Selection* window. Notice that the screen displays the filename and a description of the file, if one exists. The screen displays an additional column for the application name. When the file list screen is displayed, PlotTalk positions the cursor on the first file in the alphabetical list. If you have already used TableTalk or PlotTalk to generate a report or graph in the current FOCUS session, the last file you used is listed first. This screen lists FOCUS files or views that are currently available for reporting in an alphabetical list.

You select the one you wish to use to create a graph from the *File Selection* window. Your screen may display different files from those shown in the following image:



To select the EMPLOYEE file, use a combination of the PF7 (page up), PF8 (page down), arrow and TAB keys to position the cursor next to EMPLOYEE and press Enter. PlotTalk automatically generates the following FOCUS code in the bottom window of the screen that follows:

```
GRAPH FILE EMPLOYEE
```

Alternate Ways to Enter PlotTalk

You can also invoke PlotTalk by typing the following command at the FOCUS command prompt and pressing Enter:

```
PLOTTALK FILE filename
```

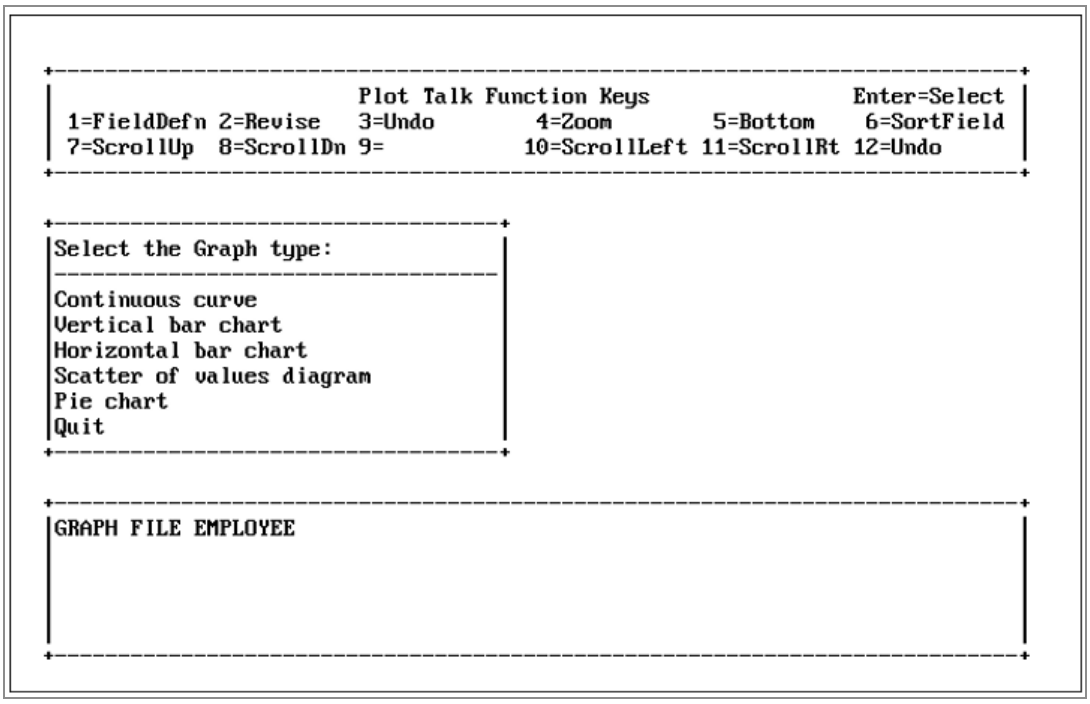
Where *filename* is the name of the file or view you want to use. This command enables you to bypass the *File Selection* window.

Other ways to enter PlotTalk include selecting it from the FOCUS Menu facility or the FOCUS Toolkit facility. Instructions for using both facilities are provided in Introducing Talk Technology.

Aborting a Request

You can abort a request and return to the FOCUS command prompt at any point in PlotTalk. Simply press PF3 until you reach the *Select the Graph type:* window, as shown in

the following image:



To exit PlotTalk and return to the FOCUS command prompt, select the *QUIT* option. The *QUIT* option is also available on the last PlotTalk window.

Creating a Graph Request

The tutorial that guides you through a PlotTalk session begins here. After you select a file, in this case EMPLOYEE, PlotTalk displays the *Select the Graph type* window.

This screen, like all screens in PlotTalk, is divided into three main areas:

- Special key information is provided in the top window. For more information on special keys, see Introducing Talk Technology.
- Selection windows appear in the middle.
- Graph requests being created are displayed in the bottom window.

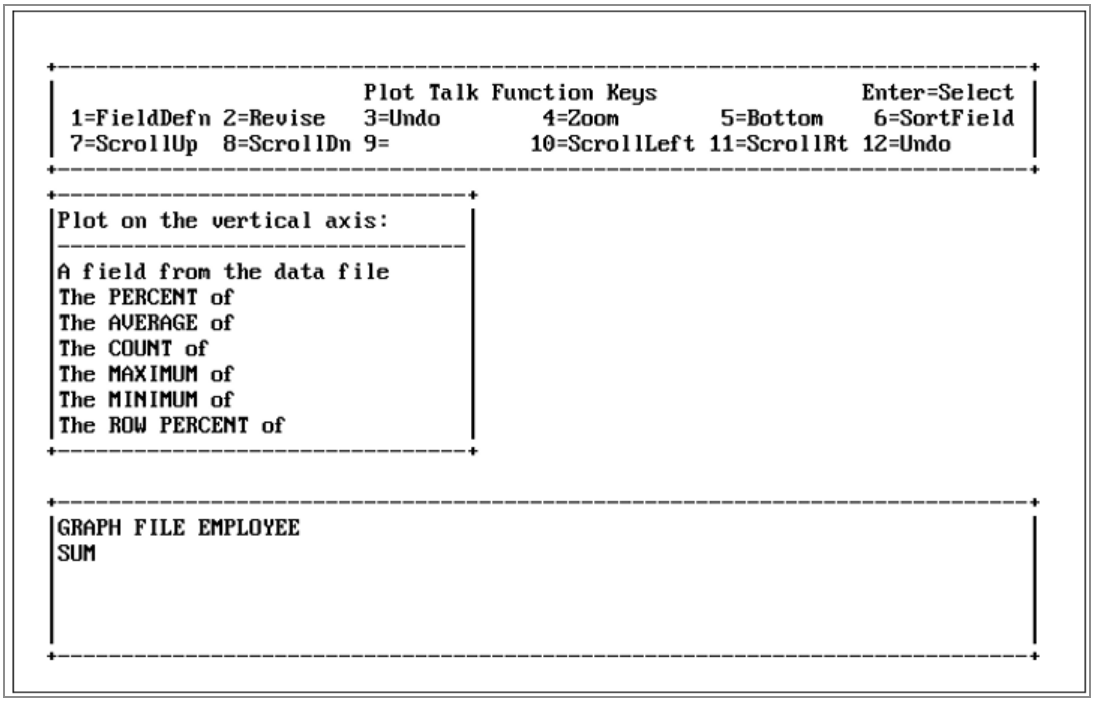
As you make selections, PlotTalk adds them to your GRAPH request in the bottom window. This request is the actual FOCUS syntax that produces the graph. If you make a mistake, press PF3 to remove your last selection and return to the previous PlotTalk prompt window.

The center window displays a list of graph types and the *QUIT* option. The graph types listed in the window depend on the range of devices available to you.

To resume the request, select the following graph type and press Enter.

```
Continuous curve
```

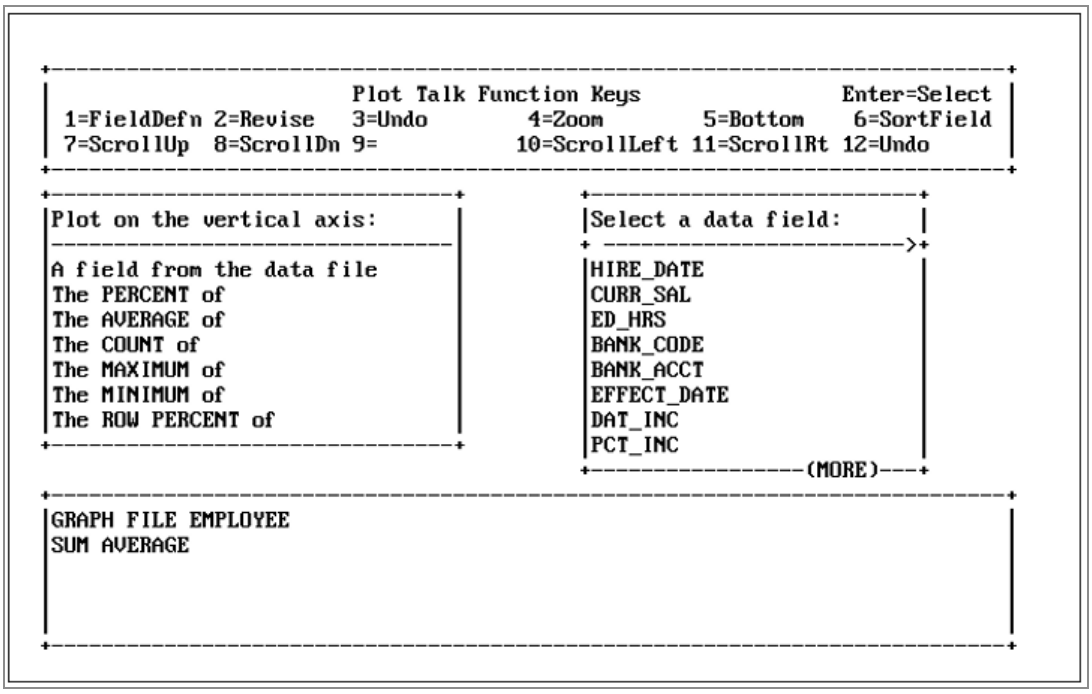
The following screen, which adds an element to the request and prompts you to *Plot on the vertical axis:*, is shown in the following image:



Now, you must select the value or function you want to appear on the vertical axis. Select the following option and press Enter:

```
The AVERAGE of
```

The following screen appears, as shown in the following image:



Entering Field Choices

You are now ready to select the field you want to average. The list of fields you see depends on the selection you make in the *Plot on the vertical axis:* window. For example, if you select *The AVERAGE of*, you can only choose fields with numeric formats. If, instead, you select *The COUNT of*, you can select fields with alphanumeric, as well as numeric formats.

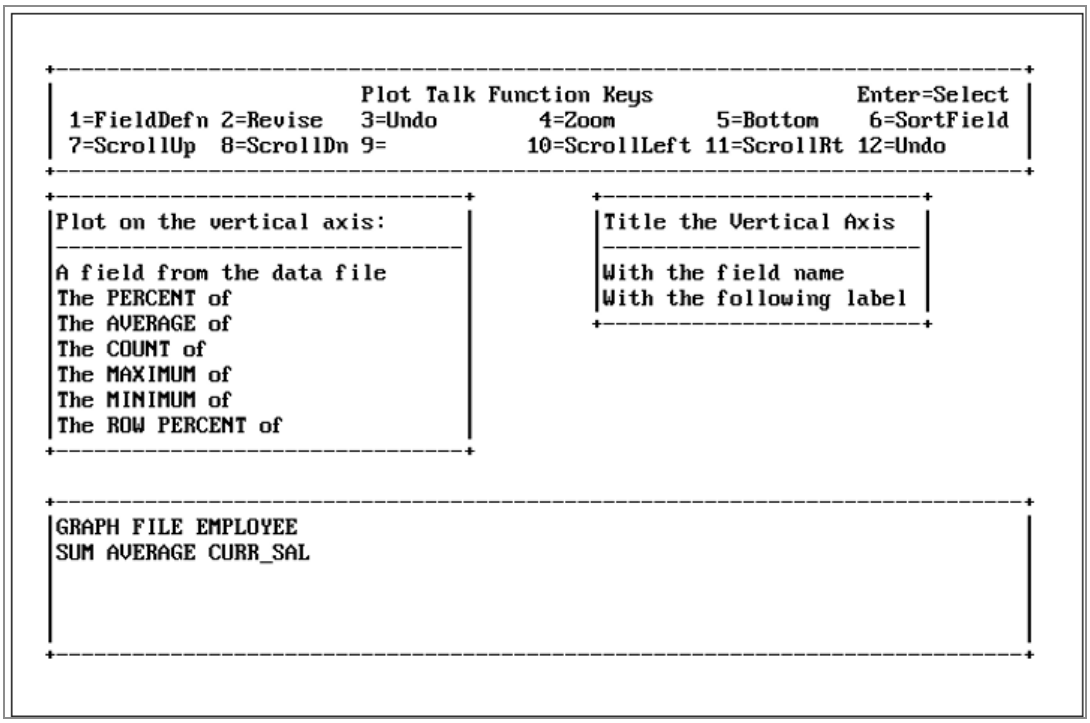
Now, average the values in the CURR_SAL field by selecting the following option:

```
CURR_SAL
```

The bottom window automatically completes the second line of GRAPH syntax:

```
SUM AVERAGE CURR_SAL
```

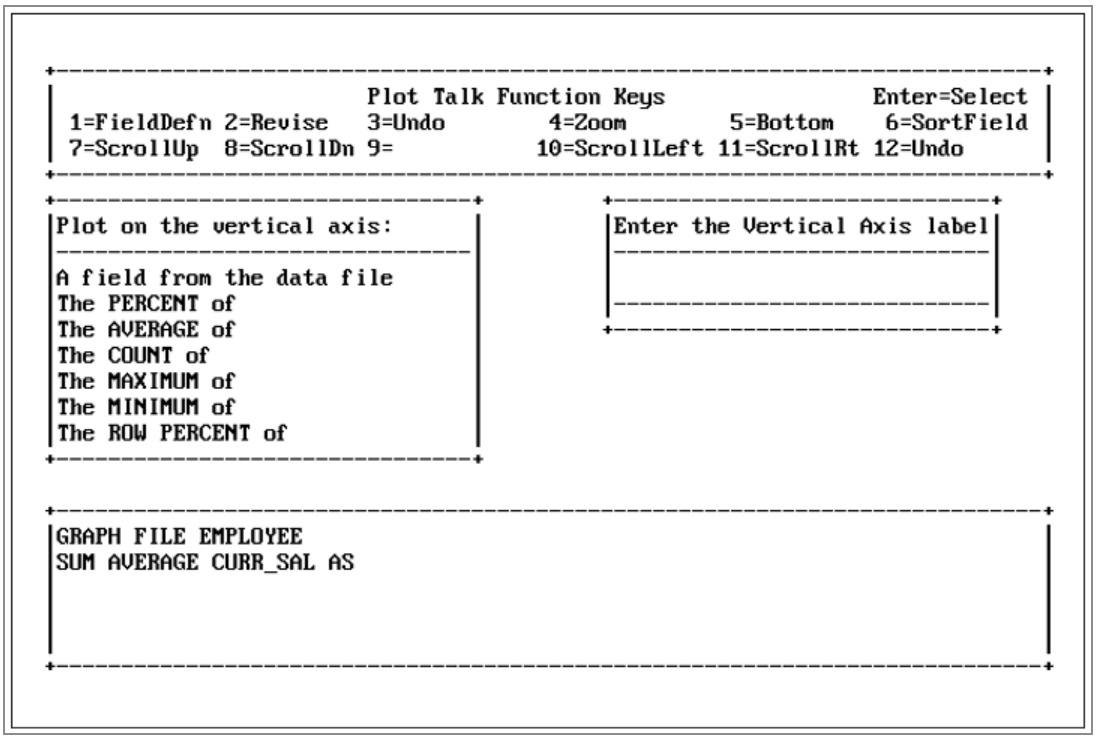
Next, the window on the right of your screen prompts you to *Title the Vertical Axis:*, as shown in the following image:



FOCUS by default uses the fieldname as the title for the axis, but you can replace the default with a label you specify. To create a label for the vertical axis, select the following option and press Enter:

```
With the following label
```

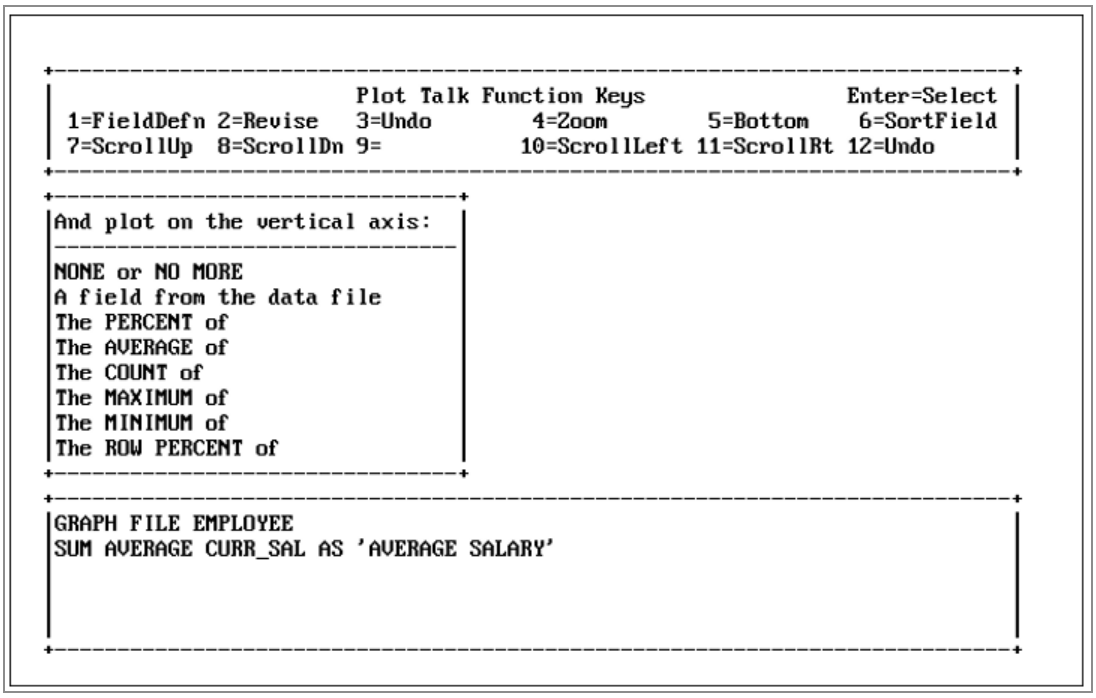
Then, you are prompted to *Enter the Vertical Axis label:*, as shown in the following image:



Type the following as the Vertical Axis label and press Enter:

```
AVERAGE SALARY
```

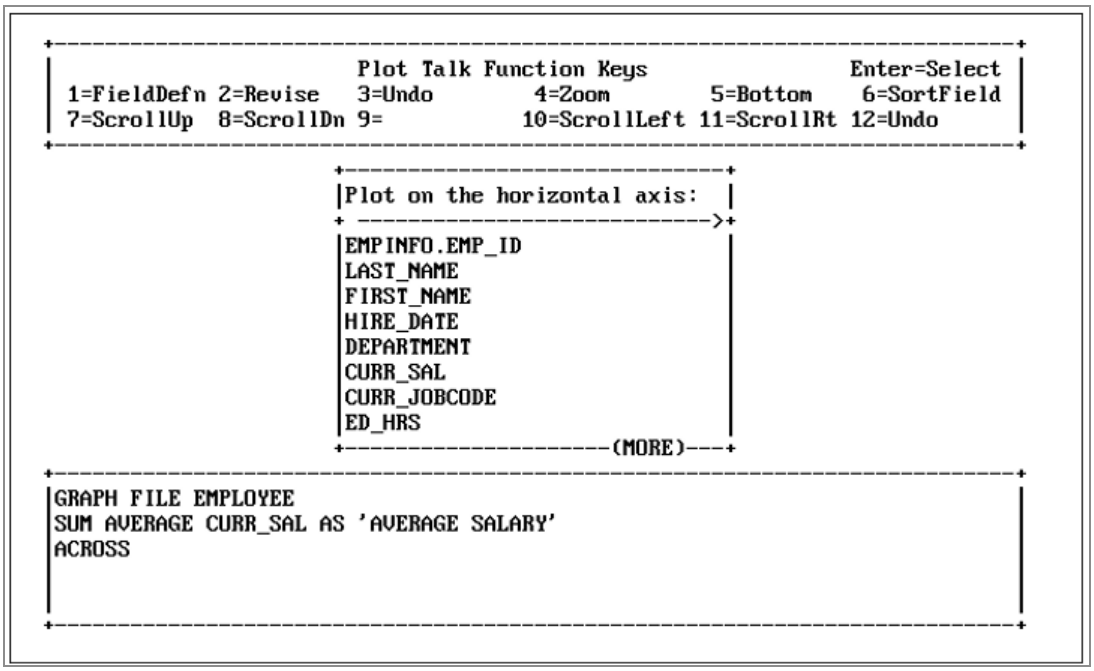
The following screen appears, as shown in the following image:



Notice that PlotTalk converts labels to uppercase and encloses them in single quotation marks ('). Now, you can plot a second field on the vertical axis or continue to the next window. To continue now to the next window, select the following option and press Enter:

```
NONE or NO MORE
```

The following screen appears and prompts you to *Plot on the horizontal axis:*, as shown in the following image:



Choose the following field and press Enter:

```
CURR_JOBCODE
```

Once again, the default label for the horizontal axis is the name of the field you specified. This time, use the default to label the horizontal axis; select the following phrase:

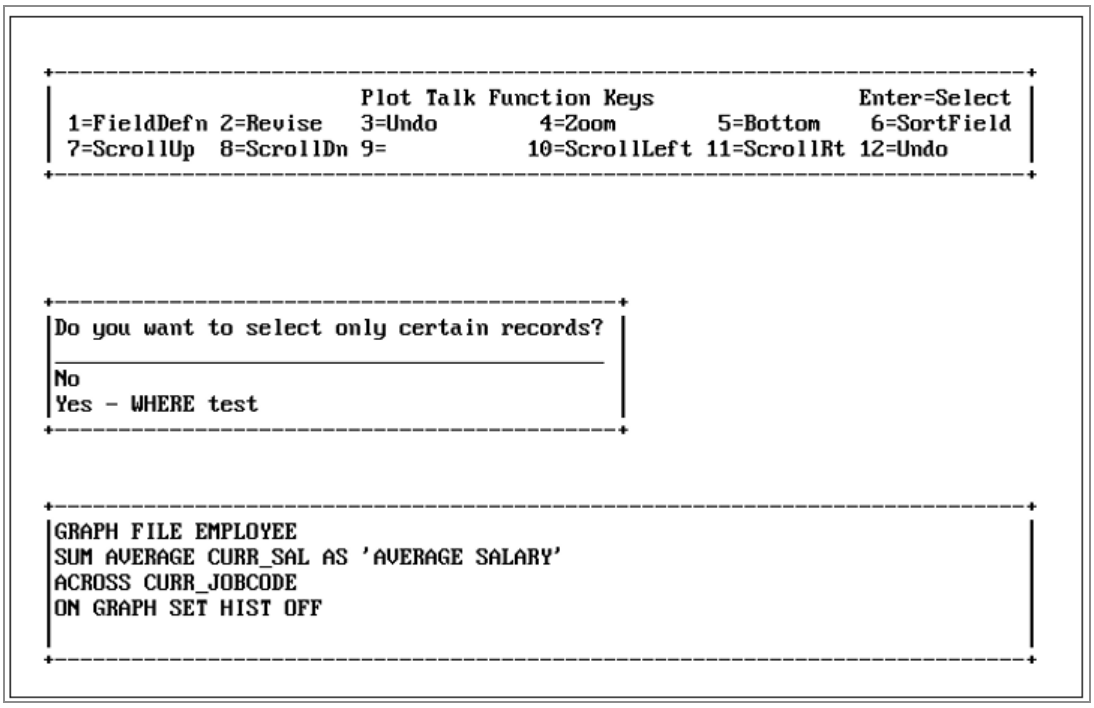
```
 With the field name
```

At this point, FOCUS inserts command so that the request prints as a continuous curve graph. If this phrase is omitted, it prints as a bar chart.

```
ON GRAPH SET HIST OFF
```

Adding Record Selection Tests

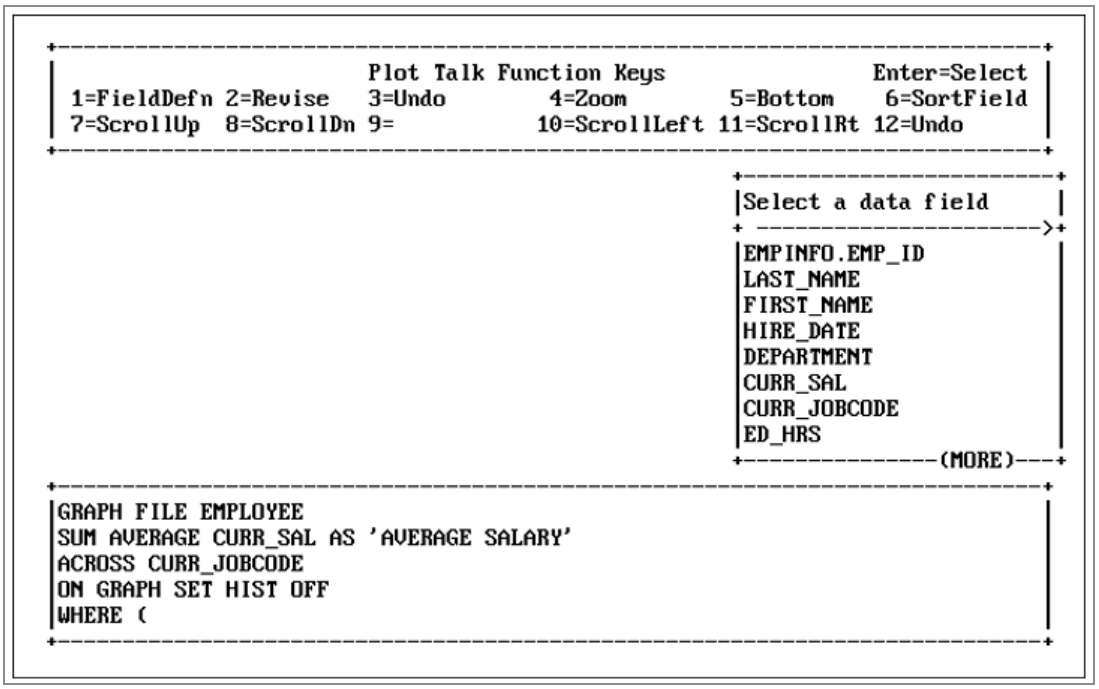
You have now selected the fields to be displayed on the graph. The windows that follow enable you to create record selection tests for your request. The next window prompts you for screening conditions, as shown in the following image:



For this example, use the WHERE clause to screen the data. Select the following option from the *Do you want to select only certain records?* window and press Enter.

```
Yes - WHERE test
```

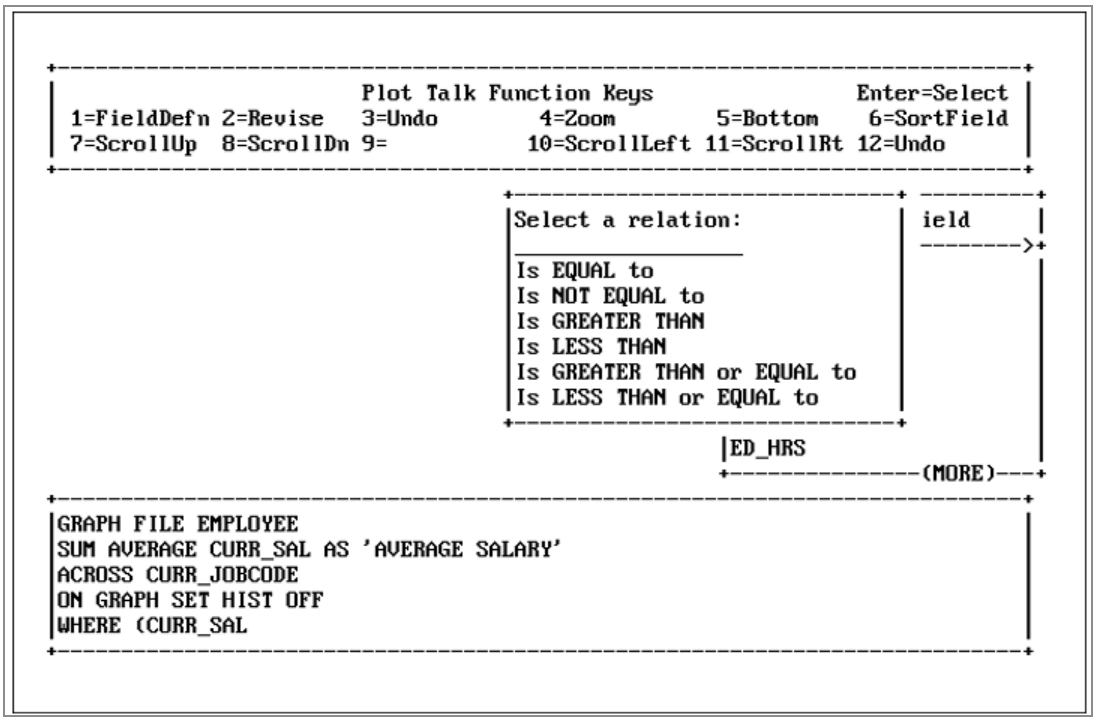
The following *Select a data field* window appears, as shown in the following image:



Select the following screening field and press Enter:

```
CURR_SAL
```

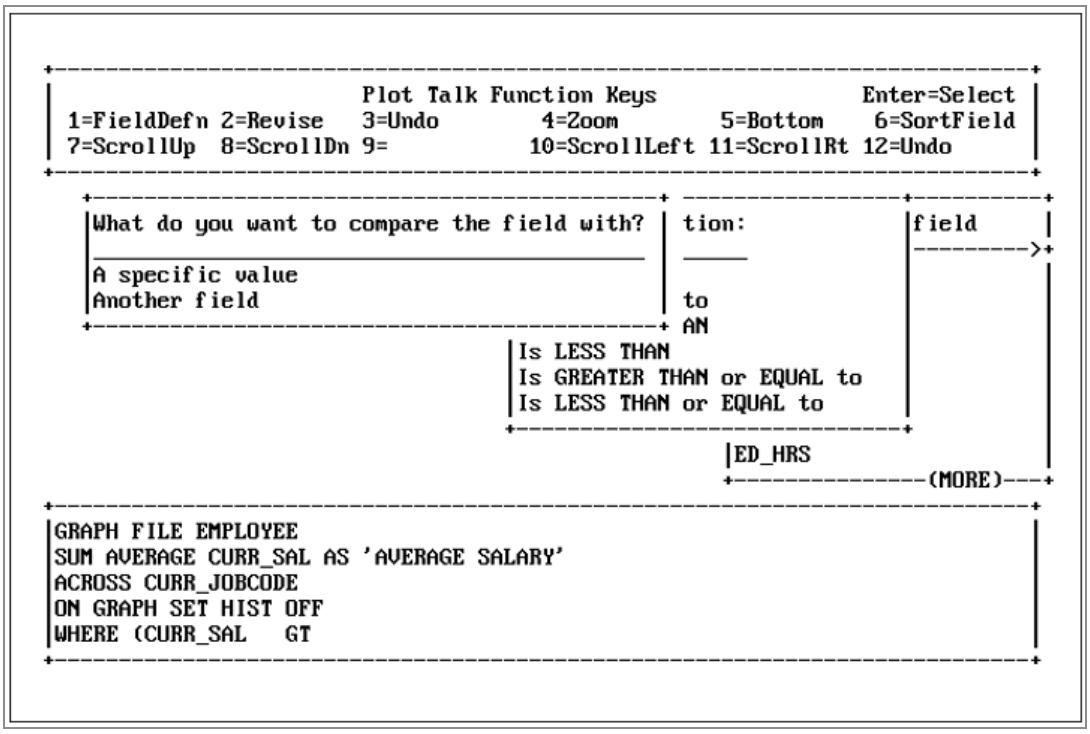
The following *Select a relation:* window prompts you for a relation, as shown in the following image:



To report on salaries greater than $15,000, select the following relation and press Enter:

```
Is GREATER THAN
```

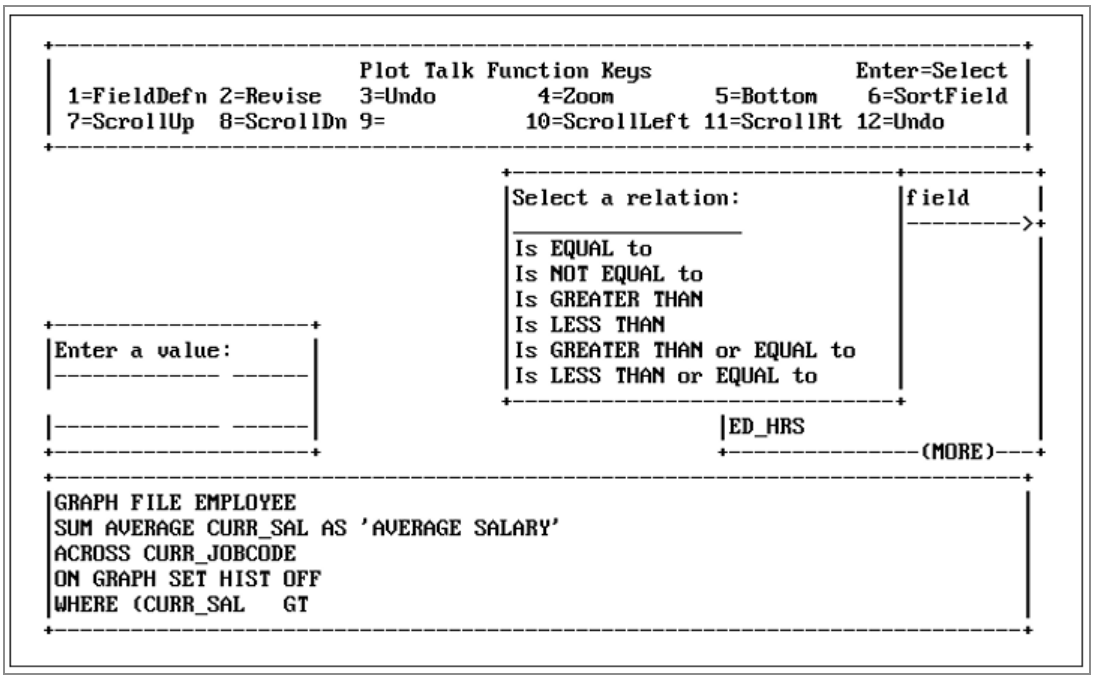
The following window appears and prompts *What do you want to compare the field with?*", as shown in the following image:



You can compare the field to a value, or to another field. Select:

```
A specific value
```

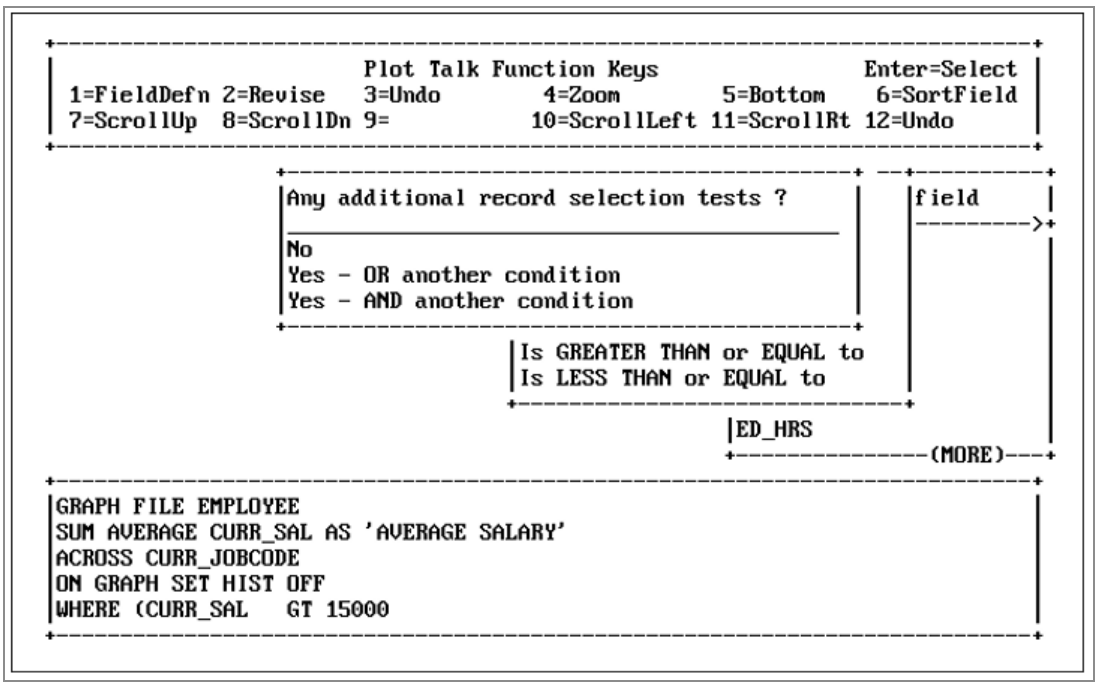
Another window appears prompting you to *Enter a value:*, as shown in the following image:



Enter the following value and press Enter:

```
15000
```

PlotTalk prompts for *Any additional record selection tests?*, as shown in the following image:



To specify that there are no additional selection criteria, select the following option and press Enter:

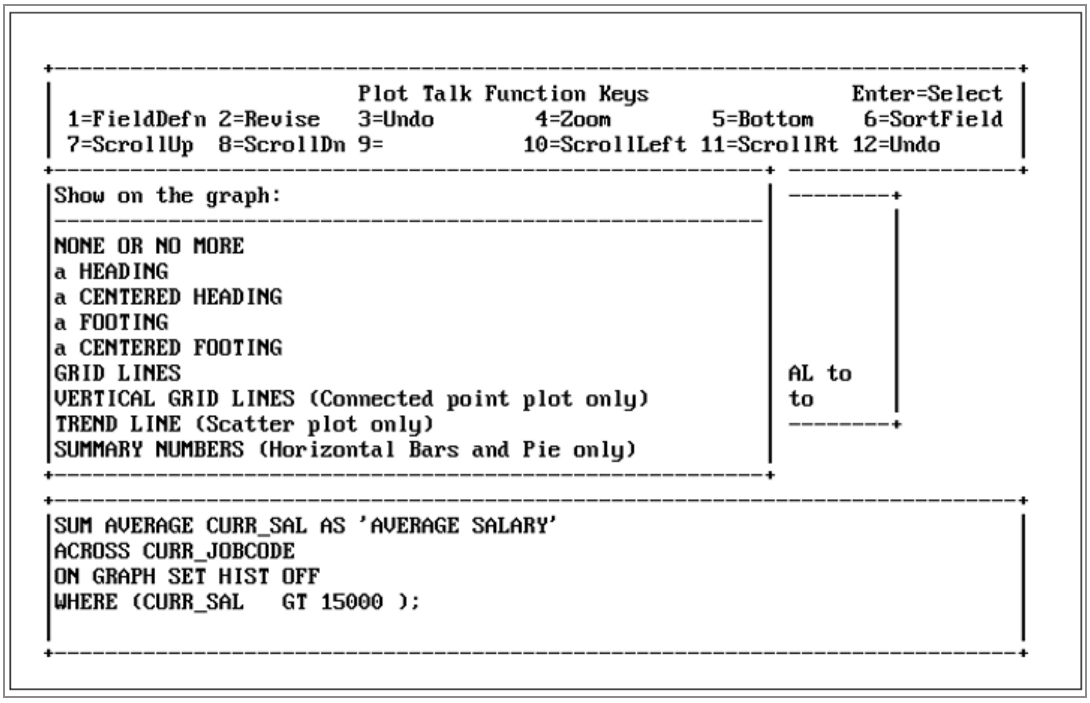
```
No
```

At this point, you have the option of specifying another WHERE condition, but for this example, select:

```
No
```

Customizing the Graph

The next window, *Show on the graph:*, contains selections that enable you to customize your output:



Select the following option and press Enter:

```
a CENTERED HEADING
```

You are prompted to *Enter your text*. Type the following text and press Enter:

```
AVERAGE EMPLOYEE SALARIES
```

This time, instead of supplying single quotation marks ('), PlotTalk includes the double quotation marks (") necessary around text for headings.

PlotTalk prompts you for *Any more text lines?*. Select

```
No
```

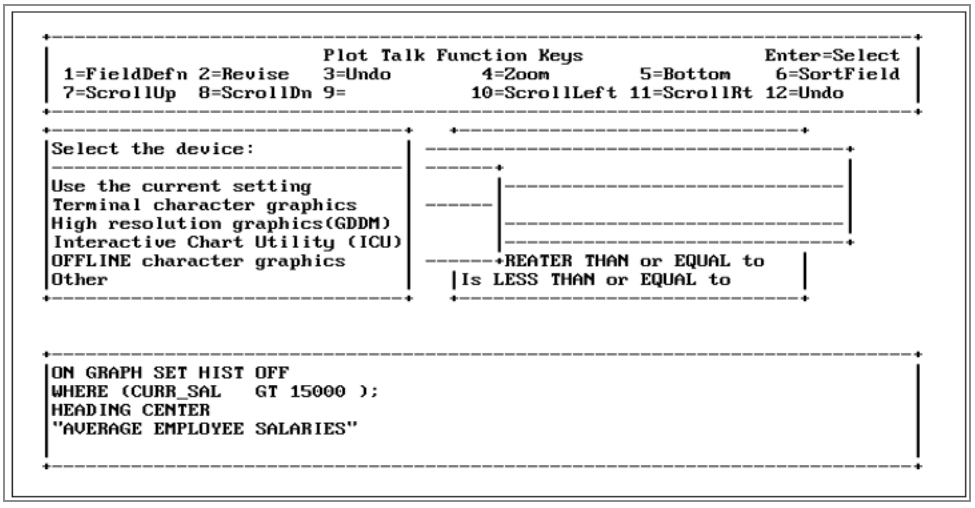
The center window *Show on the graph:* is redisplayed. At this point, you are almost finished with the GRAPH request. Select the following option and press Enter:

```
NONE OR NO MORE
```

Displaying and Printing the Graph

Next, you are prompted to *Select the device:* on which you wish to display your graph. This is the last step you take before executing the request. For this example, use the default device, select the following option, and press Enter:

```
Use the current setting
```



If you want to select an option other than the default *Use the current setting*, and if you have the appropriate configuration, you can select one of the other choices. See your

FOCUS administrator or the *FOCUS for IBM Mainframe User's Manual* for details on your PROFILE FOCEXEC configuration.

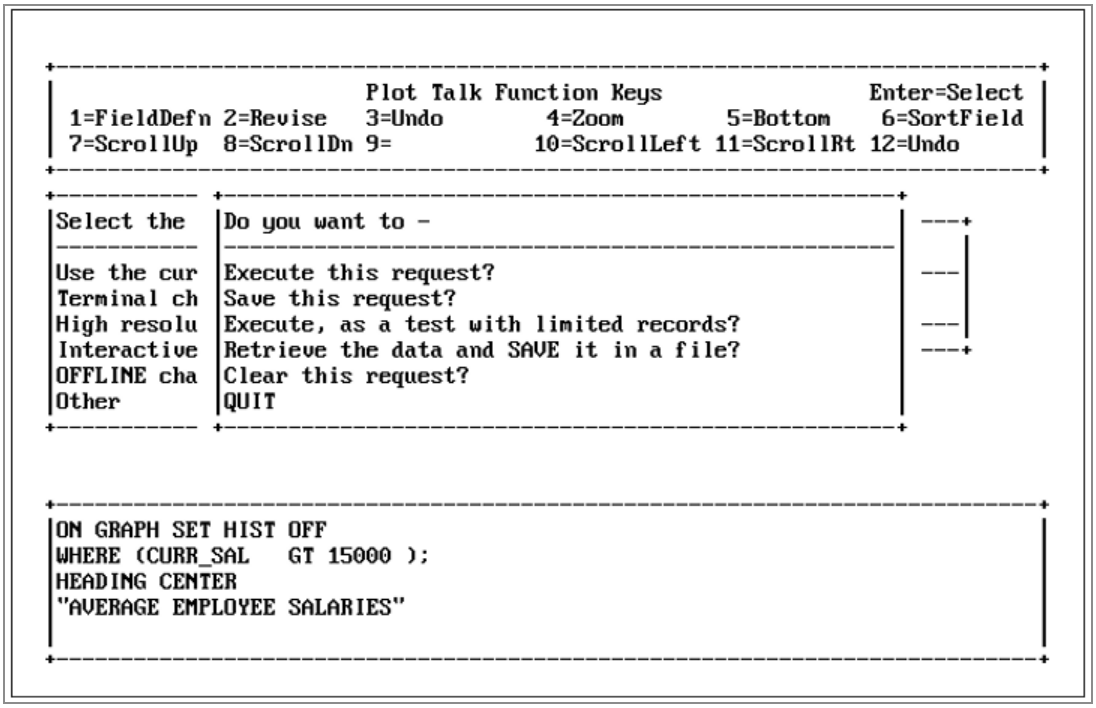
Choices you can make from the *Select the device:* window are described in the following table.

Choice	Description
`Use the current setting`	Uses the current setting of the DEVICE parameter. The default DEVICE is IBM 3270, which gives terminal character graphics. DEVICE can be changed to any device type appropriate to your configuration (for example, standard, GDDM, ICU). If your terminal is of high-resolution quality, but your current setting is for a standard type, you might want to select another option.
`Terminal character graphics`	Is the standard driver package, which is part of the FOCUS system. In some situations, this is identical to the Current Setting device.
`High resolution graphics (GDDM)`	Is available only to high-resolution IBM monitors (3179, 3279) with GDDM graphic software installed. It creates a high-quality graph. Note: GDDM or ICU users who wish to print high-resolution graphs must print them outside of the FOCUS environment. To do this, either print from within GDDM or ICU, or save from within GDDM or ICU and then print at the appropriate operating system level. Consult your FOCUS administrator and/or your *IBM GDDM Guide for Users* or *FOCUS ICU Interface Manual* for instructions.
`Interactive Chart Utility (ICU)`	Is available only to some users of GDDM systems. ICU is a menu-driven program that creates extremely high-quality graphs. The user can easily adjust graph size, type, and/or legend with ICU. Note that the ICU Interface, an optional product, is required for FOCUS users who wish to use ICU.

Choice	Description
`OFFLINE character graphics`	Sends the graph to an offline printer. Check with your FOCUS administrator for the identification of the printer.
`Other`	Displays a menu of all the possible device types, including **Hewlett-Packard® and Tektronix®** models. Select this option only if DEVICE was not properly set previously.

Executing a Request

The next step in the tutorial is to execute the request you have created. To execute a request, the *Do you want to* window must be displayed on the screen, as shown in the following image:



This window displays several choices, as described in the following table.

Option	Description
`Execute this request?`	Executes the request and produces a graph.
`Save this request?`	Saves the request as a FOCEXEC file which you can edit or execute at a later time from the FOCUS command prompt.
`Execute, as a test with limited records?`	Prompts you to enter the number of records you want to test and executes that number.
`Retrieve the data and SAVE it in a file?`	Creates the graph and then places it into a specially-formatted file called FOCSAVE. You can display the FOCSAVE file from the FOCUS command prompt at a later time. Note: This feature is available only for certain types of terminals.
`Clear this request?`	Erases the current request and starts a new PlotTalk session. To avoid losing your work, save the request before clearing it. You can generate a new request using the same file or a different one.
`Quit?`	Quits out of PlotTalk and exits to the FOCUS command prompt.

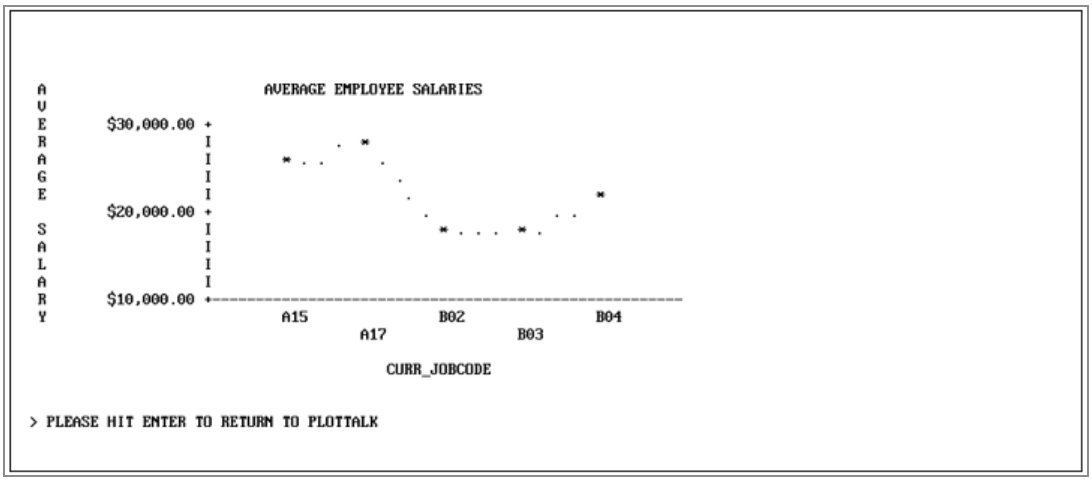
For this example, select:

```
Execute this request?
```

The next screen provides a graph summary; it displays the number of records you have selected and the number of points plotted, as shown in the following image:

```
>
 NUMBER OF RECORDS IN GRAPH=        8  PLOT POINTS=      5
 PAUSE..PLEASE ISSUE CARRIAGE RETURN WHEN READY
```

Press Enter to display the graph, as shown in the following image:



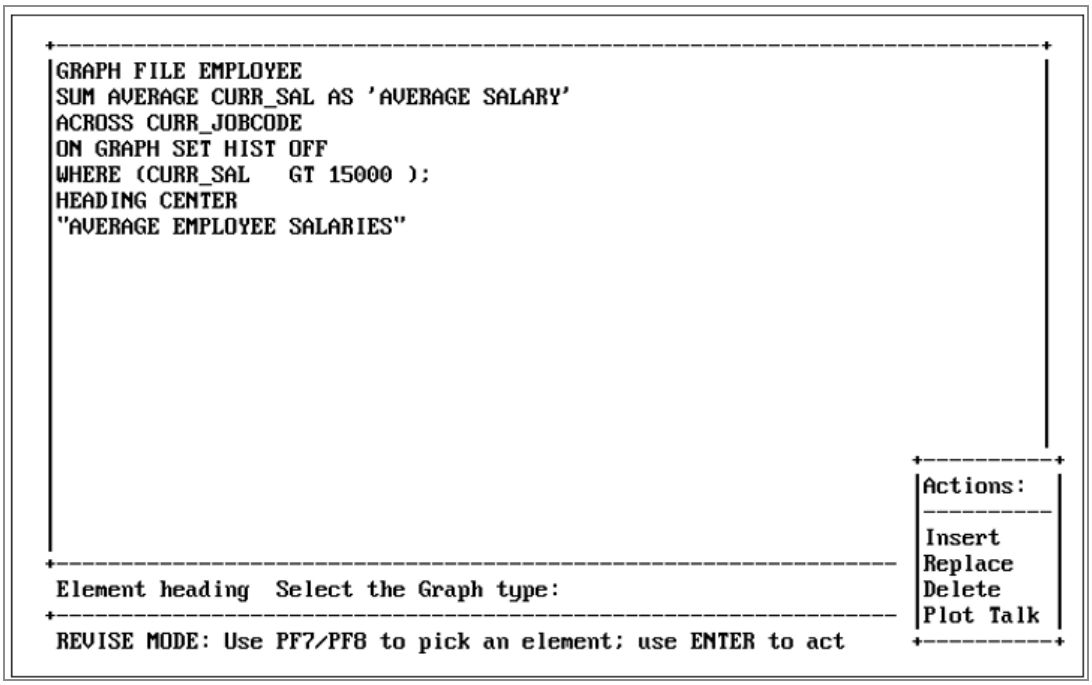
This graph shows the average salaries for every job description sorted by job code. Of course, this is only a simple example of the sort of graph you can create.

To return to PlotTalk and the *Do you want to* window, press Enter twice. You can quit at this point, but to illustrate how to revise a graph request, press PF2.

Revising Your Report Request

You can edit a PlotTalk request that you are working with at any stage by pressing PF2 to use Revise Mode.

Revise Mode displays an expanded screen with your current GRAPH request, as shown in the following image:



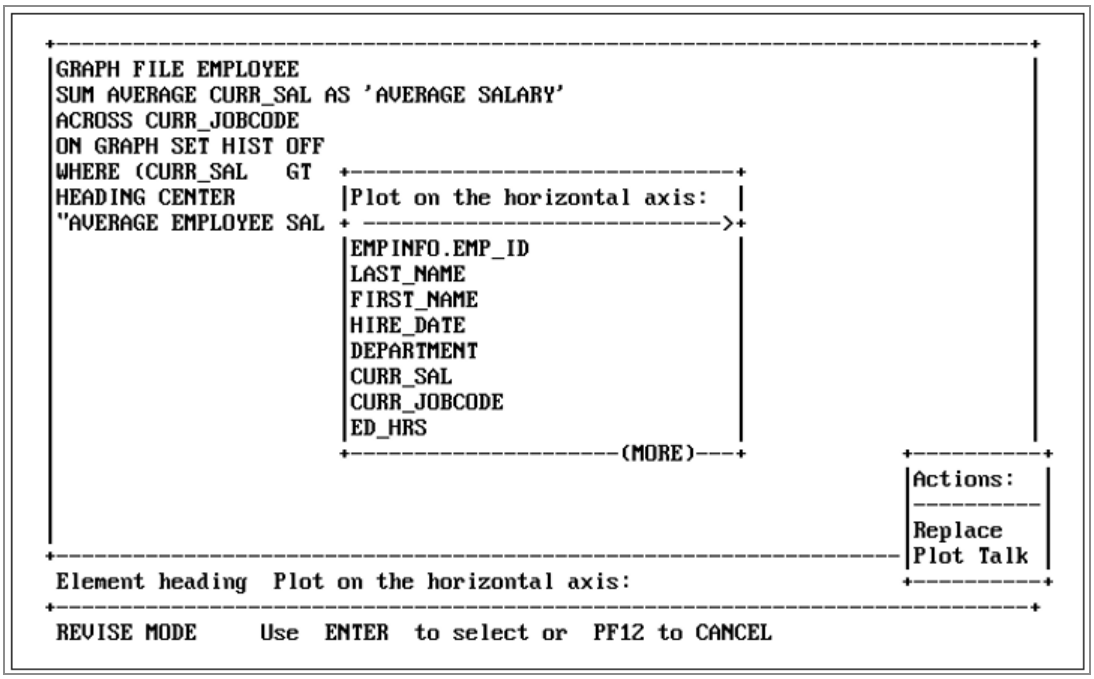
This screen consists of:

- A large center window that displays the graph request in FOCUS syntax. Notice that the word SUM is highlighted. PlotTalk highlights elements that you can change and windows from which you can make a selection. If you do not see the highlighting, adjust the contrast controls on your terminal.
- A cursor that selects the word affected by the option you choose from the *Actions:* window in the lower right corner. (Use PF7 and PF8 to highlight the request element you want to change, and the up and down arrow keys to select the editing action you want to take on the highlighted text.)
- A line at the bottom of the screen labeled *Element Heading* that identifies the highlighted word as an element of your graph request.
- An *Actions*: window in the lower right corner of the screen that displays the types of revisions you can make to the highlighted text. You can insert, replace, and delete text in your request, or you can return to the current PlotTalk session.

Revise this request now to replace the current sort field. First, highlight the current sort field, CURR_JOBCODE, using PF8, then from the *Actions:* window, select

```
Replace
```

A window, *Plot on the horizontal axis:*, appears in the center of your screen displaying a list of fields from which to choose, as shown in the following image:

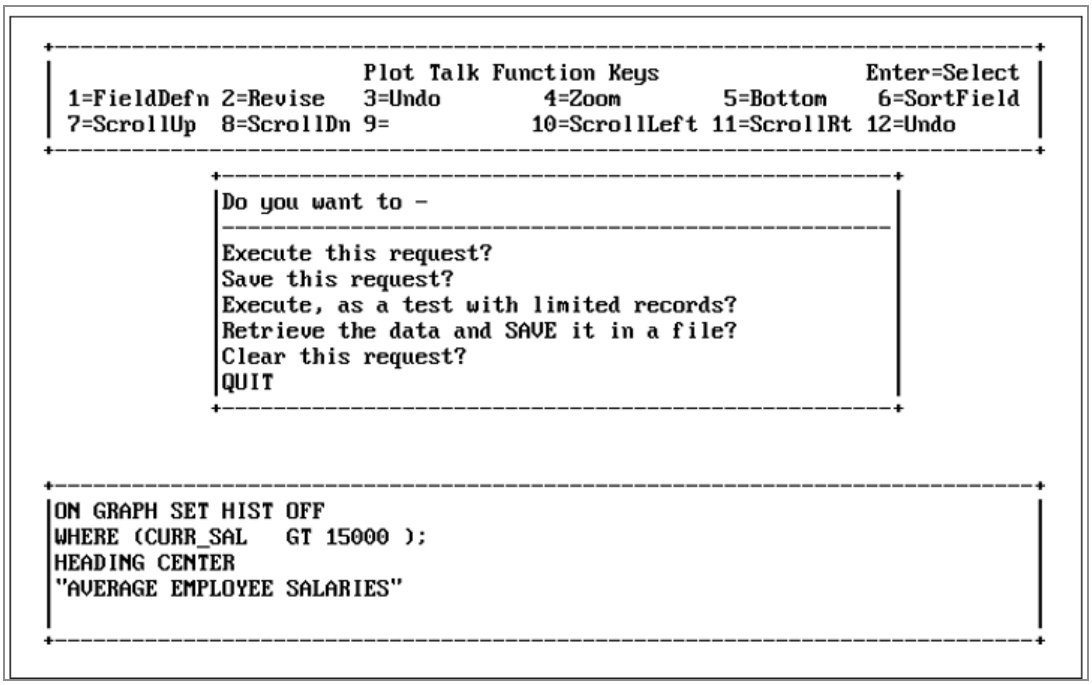


Select the following field and press Enter:

```
LAST_NAME
```

CURR_JOBCODE is replaced with LAST_NAME, which is now highlighted. You can make additional changes as desired, but for the purpose of this example, you are now ready to return to PlotTalk. To do this, you can press PF2 or select PlotTalk from the *Actions:* window.

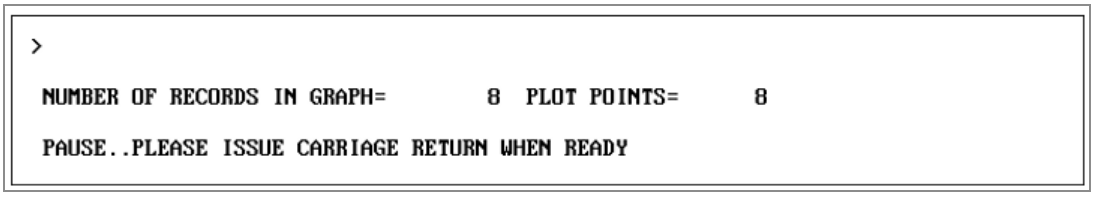
Both methods return you to the following window, so that you can execute the revised GRAPH request, as shown in the following image:



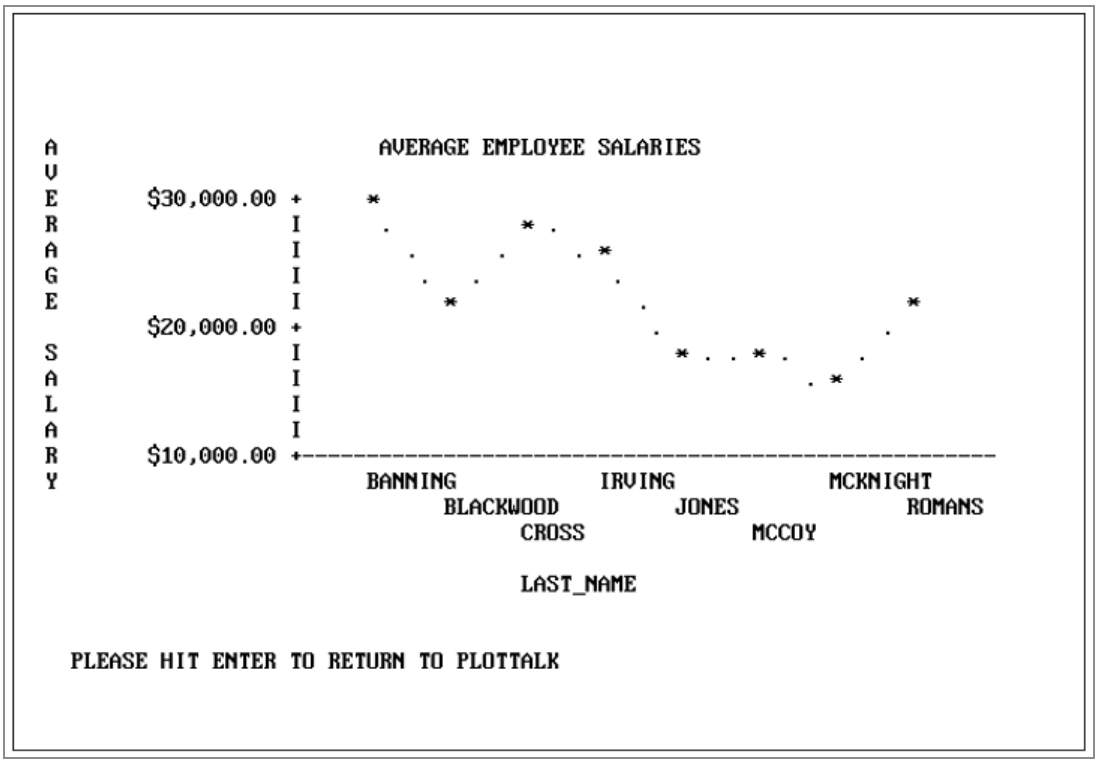
Select:

```
Execute this request?
```

Once again, the Graph Summary Screen appears and displays the number of records that were selected and how many points were plotted. This time, since you changed the request, the output is different from the output displayed for the first request you executed, as shown in the following image:



To view the graph, press Enter:



Saving the Request

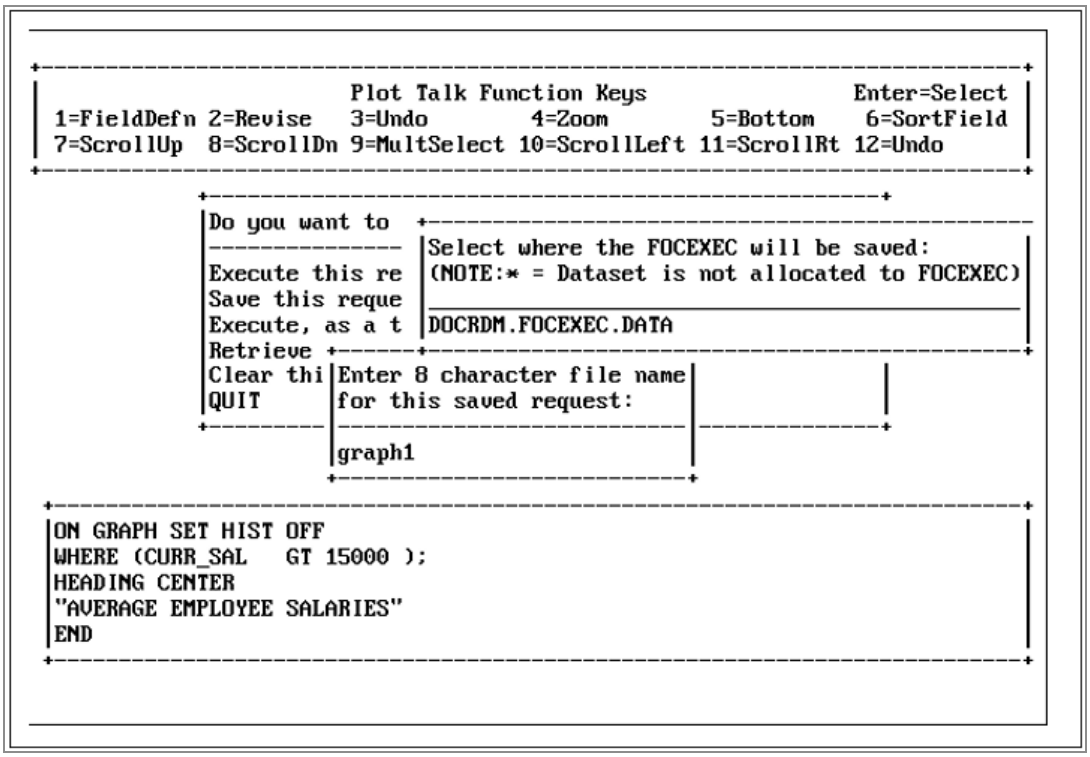
After you view the graph and press Enter to return to the *Do you want to* window, you can exit PlotTalk or perform any of the other actions described in Creating a Graph Request.

To save the request, select the following option and press Enter:

```
Save this request?
```

Type up to an eight-character name for the file and press Enter:

- Under MVS, the file is saved as a member of the partitioned dataset allocated to the ddname FOCEXEC, as shown in the following image:



Once you enter a filename, you are returned to the *Do you want to* window. To end this tutorial session, select:

```
QUIT
```

Using Help in PlotTalk

In PlotTalk, you can display user-supplied help information about files and fields. To display help in the Data Field window, move the cursor to the fieldname and press PF1. If help information is available for the field, it is automatically displayed on your screen. If you press PF1 again, PlotTalk clears the description from your screen.

Creating a HELP File With File Descriptions

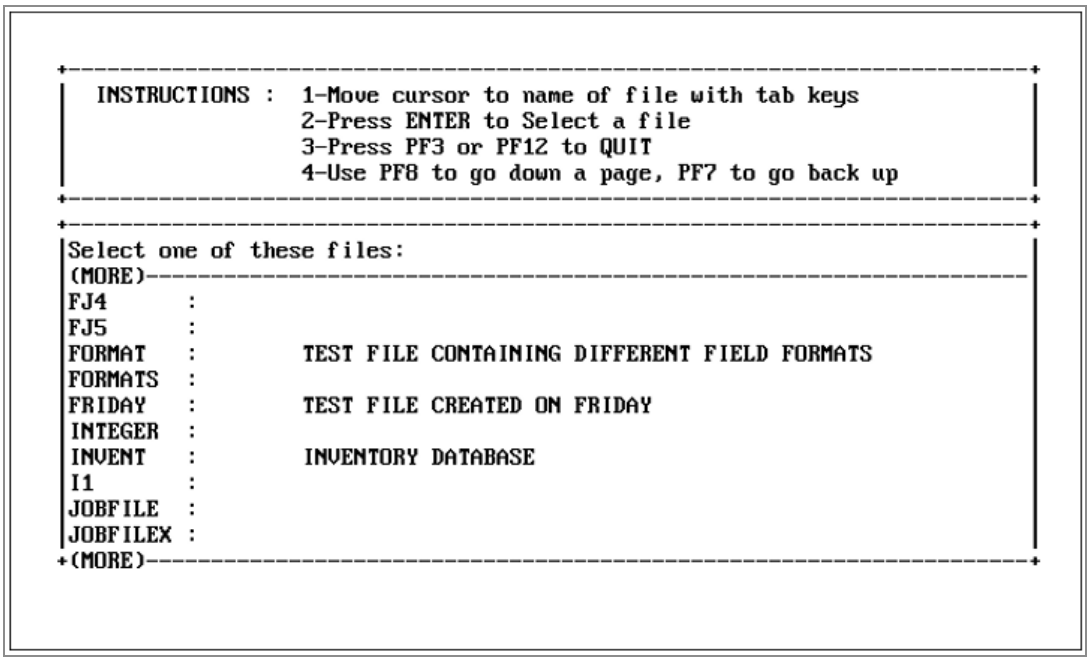
To display one-line descriptions of the files listed in the first PlotTalk screen:

- Under TSO, FOCFILES is a member in the PDS allocated to the ddname FOCDEF.

- Begin each line of the FOCFILES file with an eight-character filename in uppercase. If the filename is less than eight characters long, you must pad the remaining positions with blanks. Use the remainder of the line for the file description.

If a HELP file does not exist, the line beside the file name is blank.

The following image shows an example of a file with a file description:

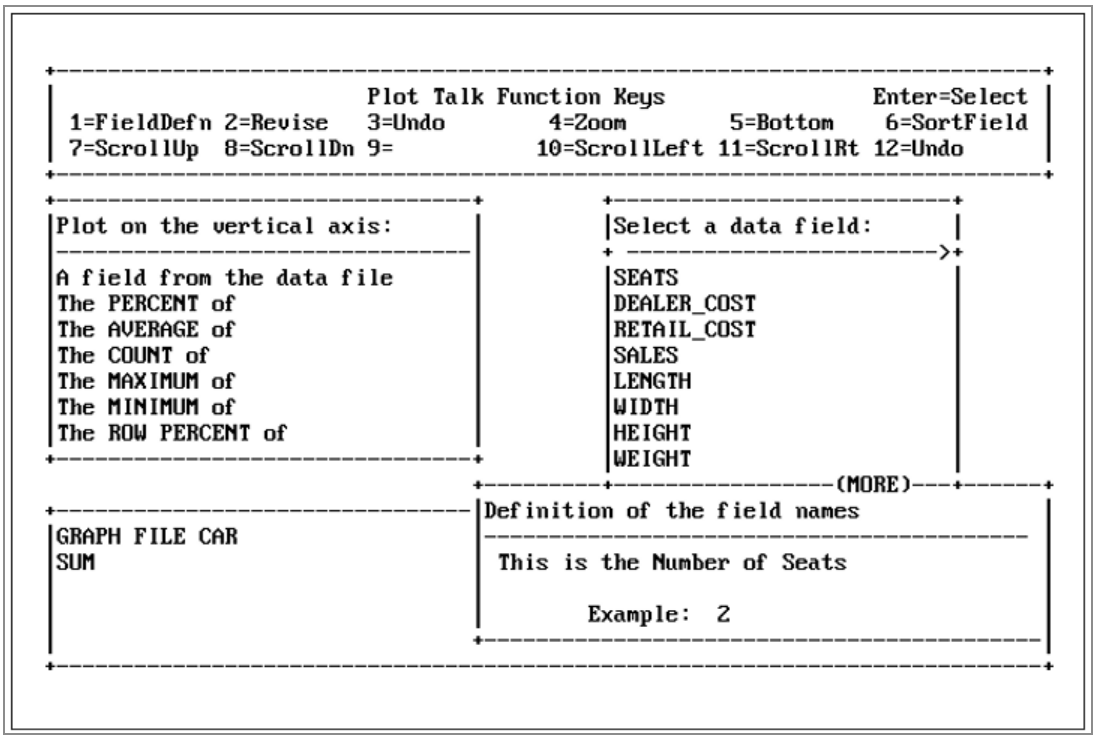


Creating a HELP File With Field Descriptions

To provide information on fields:

- Under MVS, FOCDEF is stored in a PDS allocated to the ddname FOCDEF with member names the same as the description.
- Begin each line with a 12-to-66 character field containing the fieldname in uppercase. If the field is less than 12 characters long, you must pad it with blanks.
- You can enter up to three consecutive lines of text for each fieldname.
- Enter the fieldnames in the same order as they appear in the Master File.

The following image shows an example HELP file with field descriptions:



Help information can also be created for:

- Temporary fields created with the FOCUS DEFINE command.
- Defined fields from other files that have been joined with the FOCUS JOIN command.

FileTalk

FileTalk enables you to create single-segment and multi-segment Master Files without making errors. This topic describes how you use FileTalk to create a single-segment Master File. By working through the tutorial, you learn how to create a small Master File for a file that describes the products being sold at a retail store. This file contains four items:

- Product code
- Product name
- Package size
- Unit cost

For each product code, there is one product name, one package size, and one unit cost. Because these fields are related this way, they can be logically grouped. This grouping is called a segment, and a segment corresponds to a single *table* of information in relational terms.

FOCUS and FileTalk support structured files containing multiple segments. For information about FOCUS files, see the *FOCUS for IBM Mainframe User's Manual*.

FileTalk Requirements

There are two things you should do before using FileTalk:

1. Design the information (fields) that the new database contains. Your design should be based on how you intend to use your data. The file you create with FileTalk is called a Master File in FOCUS.

   FileTalk automates the creation of this file, so you do not need to learn the file description language beforehand to create a new database. **However, more information about the file description language is provided in Appendix A, Overview of the Master File.**

2. Plan some test data for the file you create. You can use automatic data entry (AUTOMOD) screens, for single-segment files only, or you can create your own data entry procedure using the FOCUS MODIFY language. ModifyTalk can generate these

MODIFY procedures for you. (For more information on ModifyTalk, see ModifyTalk.)

FileTalk is designed to automate the creation of files, and acquaint you with FOCUS on an elementary level. Accordingly, there are certain limitations you should consider:

- FileTalk can construct new Master Files, but it cannot revise completed and saved descriptions. To make changes to completed descriptions, for example, to add TITLE, HELP, or ACCEPT attributes, among other features, use TED or your system editor. For information on these features, see the *FOCUS for IBM Mainframe User's Manual*.
- If you have revised a Master File after data has been added to the database, use the FOCUS REBUILD facility. For more information, see the *FOCUS for IBM Mainframe User's Manual*.

Basic Elements of a Master File

The basic elements of a Master File are:

- File declaration
- Segment declaration
- Field declaration

The file declaration names your Master File. The name for the Master File you use in this tutorial is PRODUCT.

The segment declaration determines the names and number of segments in your Master File. The PRODUCT Master File you create has one segment, and FileTalk automatically names this segment ONE.

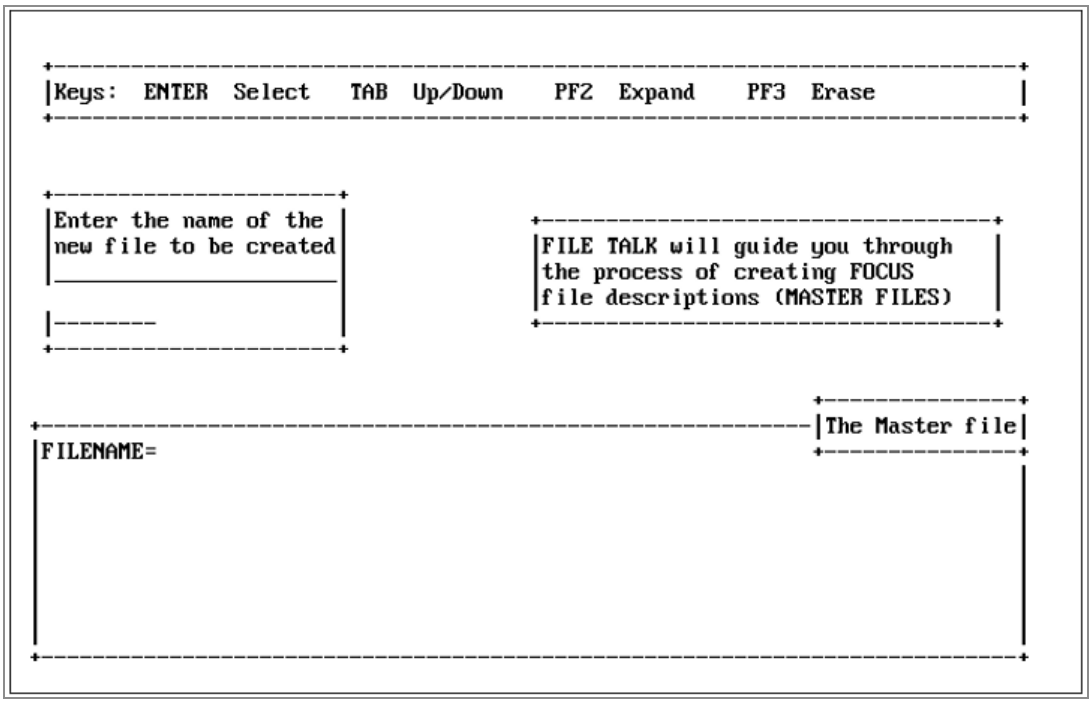
The field declaration gives each item a name, an alternate name (alias), and a format. The PRODUCT Master File will have four fields called: PROD_CODE, PROD_NAME, PACKAGE, and UNIT_COST.

Entering FileTalk

To invoke FileTalk, type the following command at the FOCUS command prompt and press Enter:

```
FILETALK
```

The following screen appears and prompts you to *Enter the name of the new file to be created*, as shown in the following image:



Subsequent screens prompt you for the segment name, and the fieldnames you want to include in your Master File.

The FileTalk screen is divided into three main areas:

- Special key information is provided in the top window. For more information on special keys, see Introducing Talk Technology.
- Selection windows appear in the middle.
- Master File being created is displayed in the bottom window.

Note:

- If you make a mistake, press PF3 to erase your previous selection.
- If you want to view more lines of a long Master File, press PF2 at any point in your FileTalk session. FileTalk displays your description in Expanded Mode. To return to FileTalk, press PF2 again.
- Another way to enter FileTalk includes selecting it from the FOCUS Menu facility.

Instructions for using this facility are provided in Introducing Talk Technology.

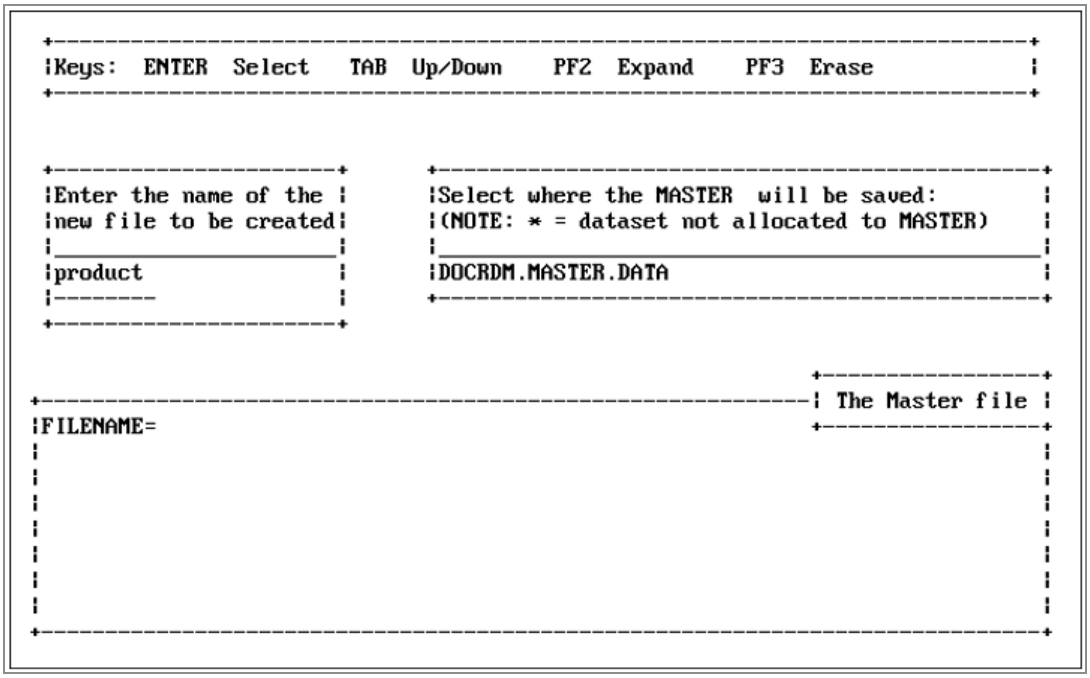
Naming the File (File Declaration)

The first step in creating a new file is to give it a name. Filenames can be up to eight characters long, and they cannot contain embedded blanks or special characters. (**For more information on file naming conventions, see Appendix A, *Overview of the Master File Description***.)

For this example, enter the following filename and press Enter:

```
PRODUCT
```

Under MVS, FileTalk displays the following additional screen in which you select the partitioned dataset. If the *prefix.MASTER.DATA* PDS is already allocated, the screen displays it, as shown in the following image. To verify the PDS, press Enter.



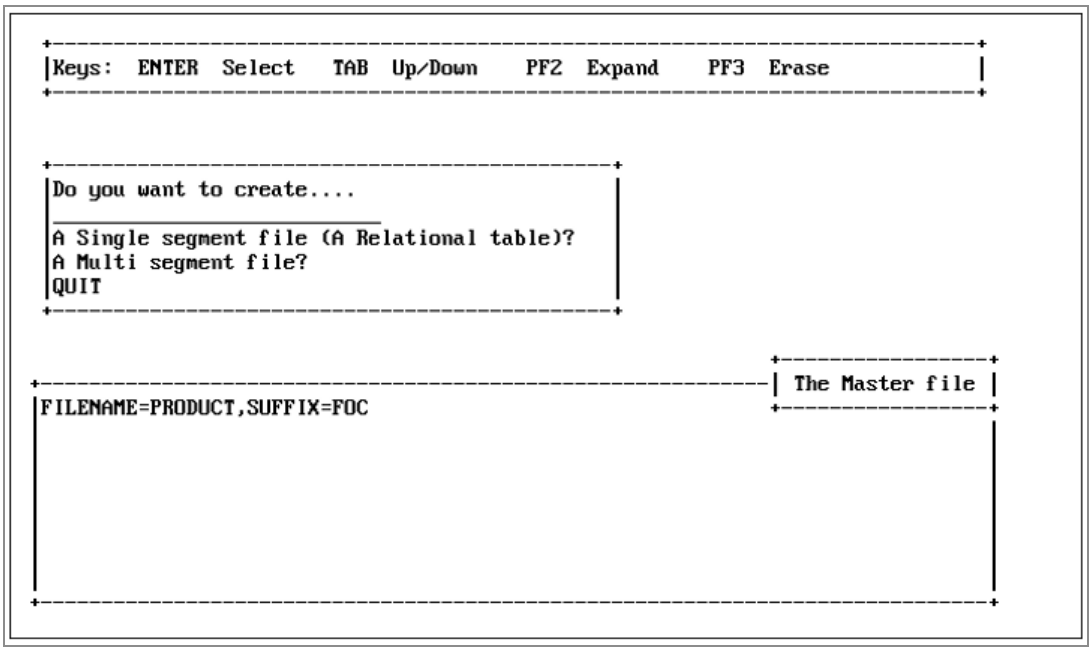
Once you provide a membername, FileTalk displays a list of partitioned datasets allocated to ddname MASTER where DISP=OLD or NEW. You can indicate the dataset to which the procedure should be saved.

If no such partitioned datasets are available, FileTalk looks for the dataset *prefix.MASTER.DATA*, where *prefix* is either your MVS user ID, or the user ID set by the

FOCUS SET PREFIX command. See the *FOCUS for IBM Mainframe User's Manual* for information on this command.

If none of these partitioned datasets is available, FileTalk writes the procedure to a temporary sequential file. This file is erased at the end of your session, unless you take the appropriate steps to save it.

The following screen displays the name of the file, as well as the type of file you are describing. The suffix is FOC, which means this is a FOCUS file.



Entering a Previously-Defined Filename

When you enter a name that has been specified already, FileTalk offers you the following three options:

- Enter a different filename
- Replace the current description
- Quit

If you select *Replace the current description*, the existing description is erased and replaced with the new one you create using FileTalk. (**Note:** Do not change the Master File for a file if you have already added data to that file.)

- Under MVS, FileTalk searches all the members of the partitioned datasets allocated to MASTER and HOLDMAST. If the filename exists and it is in a partitioned dataset to which you cannot write, you must choose a different name.

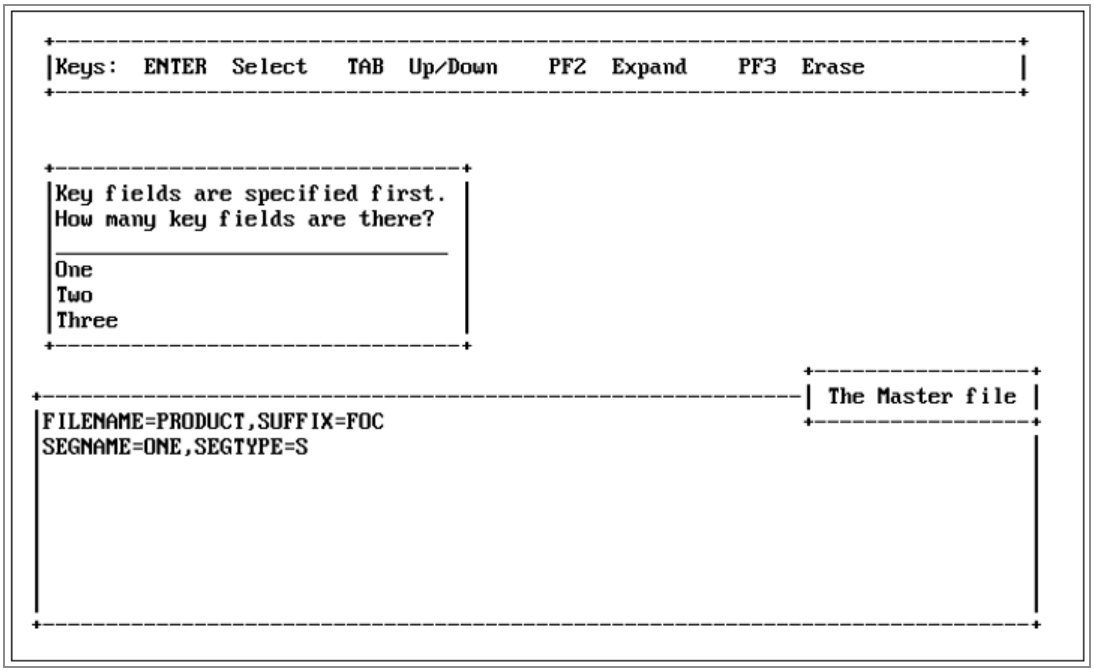
Deciding on Structure (Segment Declaration)

FileTalk enables you to create simple single-segment databases or complex multi-segment databases, such as hierarchies and networks.

The window on your screen prompts you for the structure of your Master File. Select the following option and press Enter:

```
A Single segment file (A Relational table)?
```

The following screen appears:



Notice, in the bottom window, that FOCUS has automatically assigned a name to the segment you are creating (SEGNAME=ONE). When you begin to add product data to your file, FOCUS creates an instance of this segment for each product you enter. Accordingly, FileTalk assumes that data within the file is sorted in low-to-high order (SEGTYPE=S) when creating a single-segment file. When you create a multi-segment structure, you are given the choice of sorting in low-to-high or high-to-low order.

As your FOCUS applications become more complex, the information you enter into a database may need to be organized into many logical groups. If you need to create a multi-segment file, you must specify some additional information about each segment. If you want to learn more about multi-segment files, see the following subsection. But, if you want to continue creating the sample FOCUS file, skip to Specifying Key Fields.

Creating a Multi-Segment File

Multi-segment files express a one-to-many relationship between the fields of each segment. Suppose, for example, that our PRODUCT file also tracks monthly sales amounts for each product. For this situation, you specify a second segment for the monthly sales figures.

So, when FileTalk prompts you for the type of file you want to create, select the following option and press Enter:

```
A Multi segment file?
```

The windows to follow automatically prompt you for:

- Segment names
- Segment fields
- Parent-to-child segment relationships
- Key fields
- Sort orders (low-to-high or high-to-low) for the instances in each segment.

For more information on segment naming conventions, **see Appendix A, *Overview of the Master File Description***.

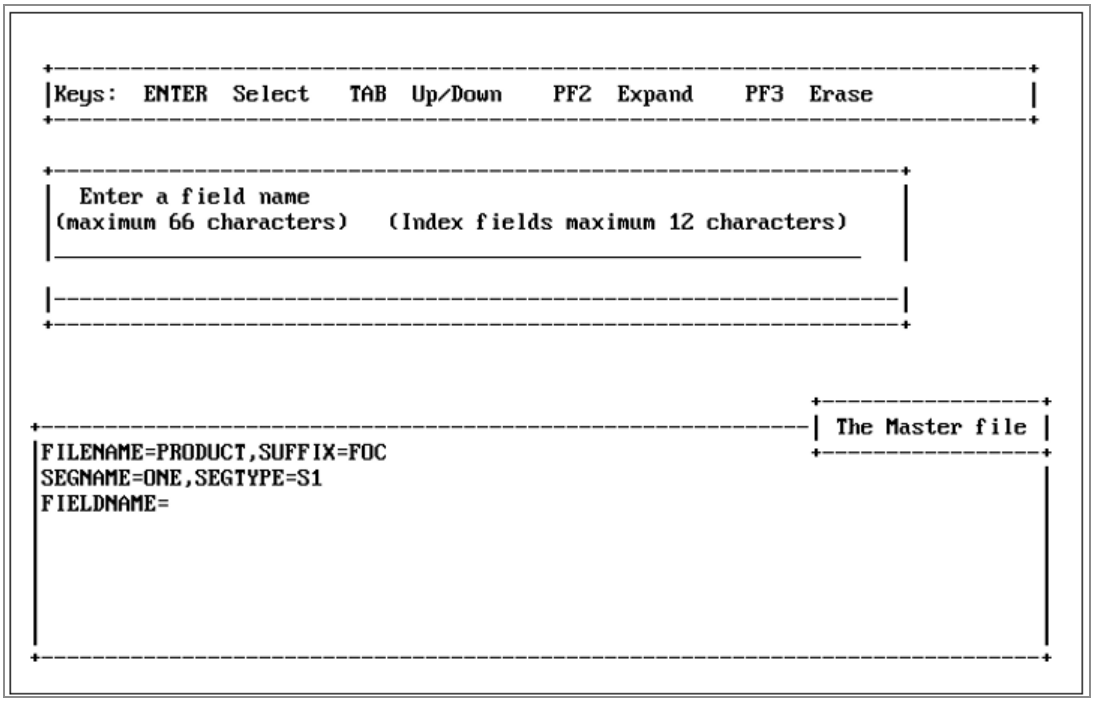
Specifying Key Fields

After you select a file structure (single-segment in this case), FileTalk prompts you for the number of key fields in the current segment. Since key fields uniquely identify a record in the database, FOCUS uses this field as the sort key to keep the file in order. Here, you can specify up to three key fields for your Master File, but if you require additional key fields, you must create or change your Master File using TED or your system editor.

PRODUCT has only one key field, PROD_CODE. Since the cursor is on

```
One
```

press Enter, and FileTalk displays the screen shown in the following image.



You are prompted to *Enter a field name* with a maximum of 66 characters. Also, notice in the bottom window that FileTalk has changed the SEGTYPE from S to S1. This means the instances are sorted in low-to-high order based on the data values in the first field.

Specifying Fields (Field Declaration)

The next set of windows in FileTalk prompts you for information about the fields in each segment. Specifically, they prompt you for the field name, alternate name (alias), format, and whether or not it is an index in your Master File.

Naming Fields and Providing Aliases

Enter the fieldname for the first field in the segment. Remember to specify key fields first. Type the following fieldname and press Enter:

```
PROD_CODE
```

Note that fieldnames are the default headings in your report, so it is recommended that you use names that accurately describe the field content. Fieldnames can have up to 66 characters, with the exception of indexed fields, which are limited to 12 characters.

Next, you are prompted to *Enter an optional alternate name for this field*. The alternate name, or alias, can be an abbreviation of the fieldname or a different name you specify and can also be a maximum of 66 characters.

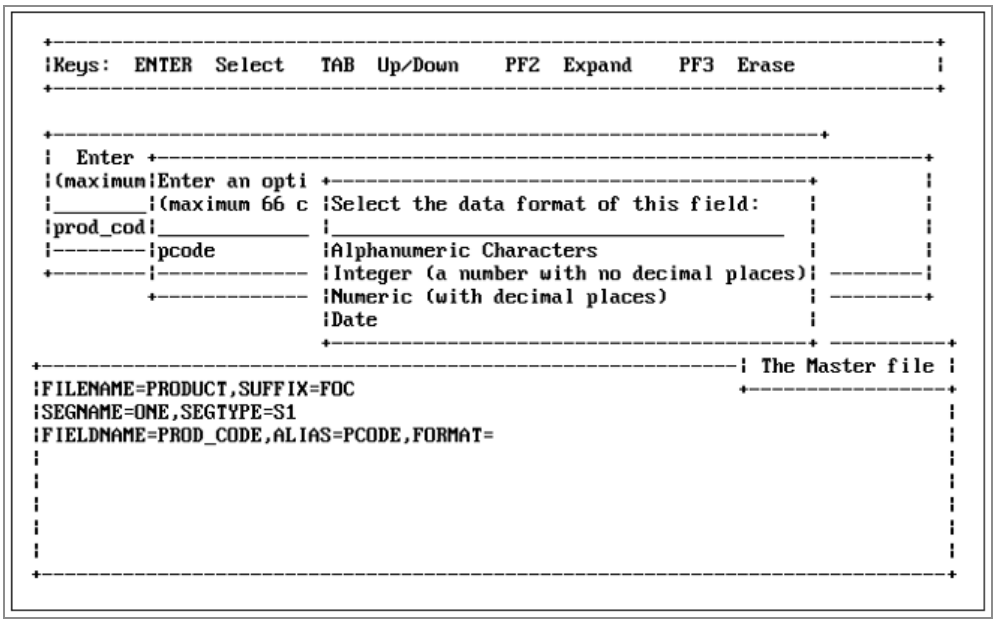
Type the following alternate name (alias(for PROD_CODE and press Enter:

```
PCODE
```

For more information about field naming conventions, **see *Appendix A, Overview of the Master File Description***.

Defining Field Formats

After specifying a name and an alias for your field, FileTalk prompts you for its data format, size, and display characteristics, as shown in the following image:



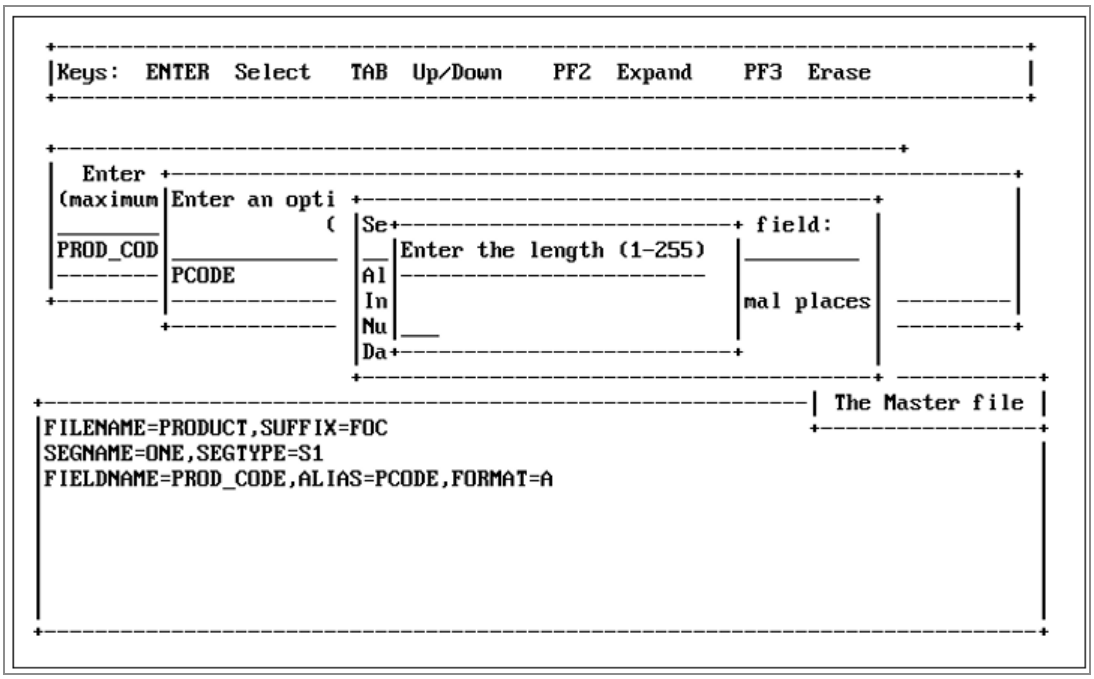
Before you select a format for PROD_CODE, here is a brief discussion and illustration of format selections you can make.

Fields can have the following data types:

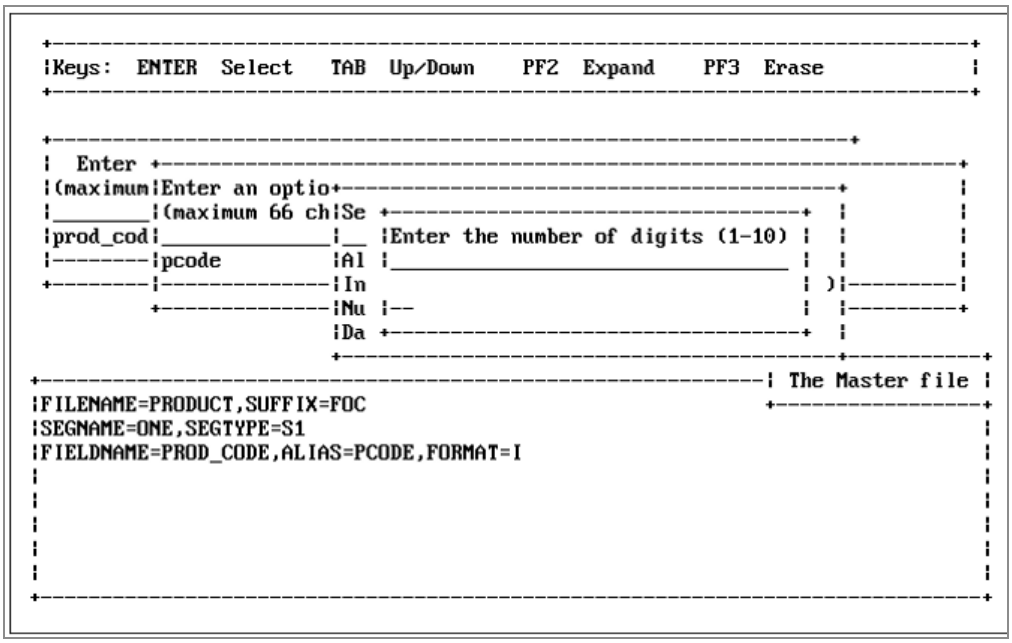
- Alphanumeric
- Integer
- Numeric
- Date

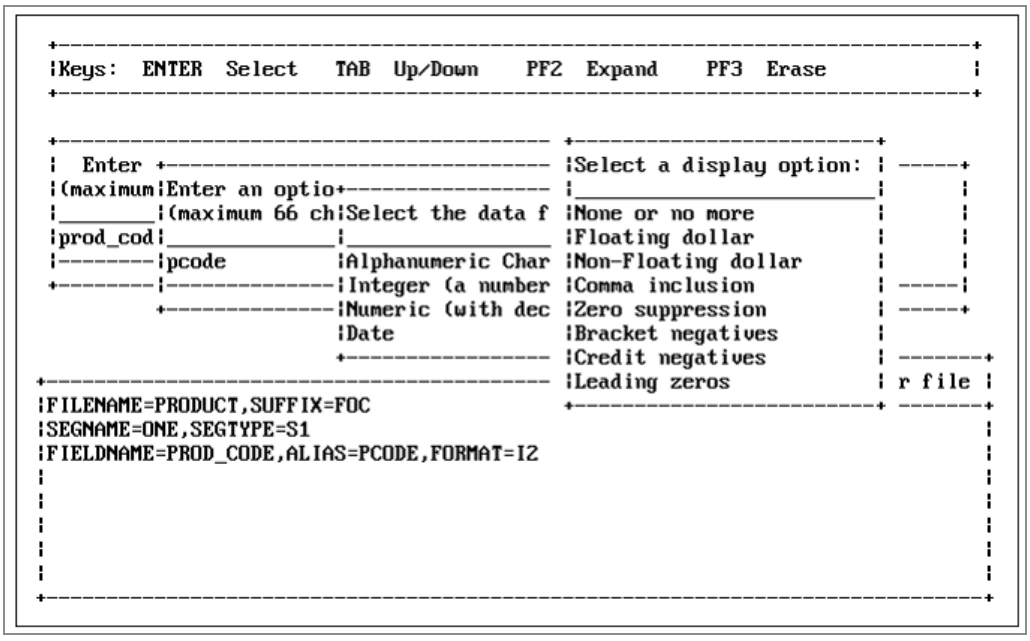
The following screens show how each data type is selected.

An alphanumeric field consists of one to 255 numeric and alphabetic characters. For example, a street address, text, or code composed of both letters and numbers. If you select this option, FileTalk prompts you for the field length, as shown in the following image:

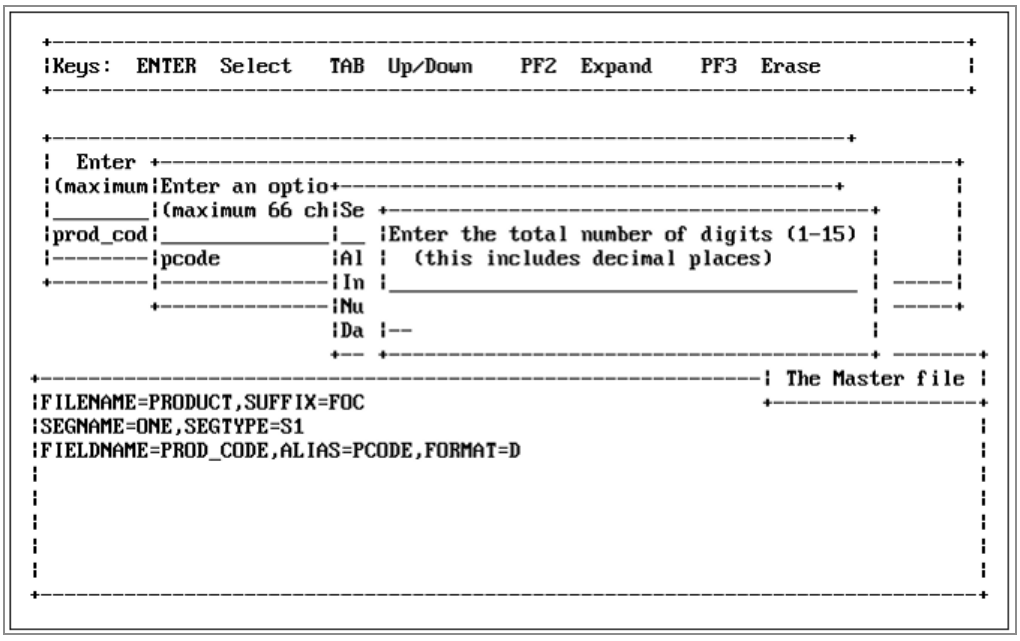


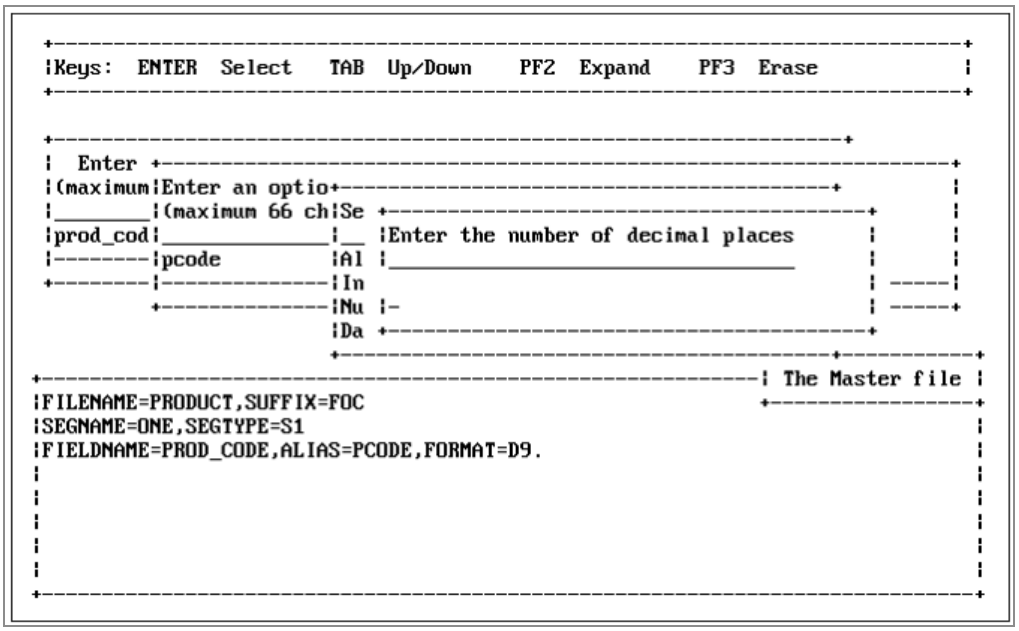
An integer field consists of one to nine digits with no decimal places. If you select this option, FileTalk prompts you for the field length and display options, as shown in the following two images:



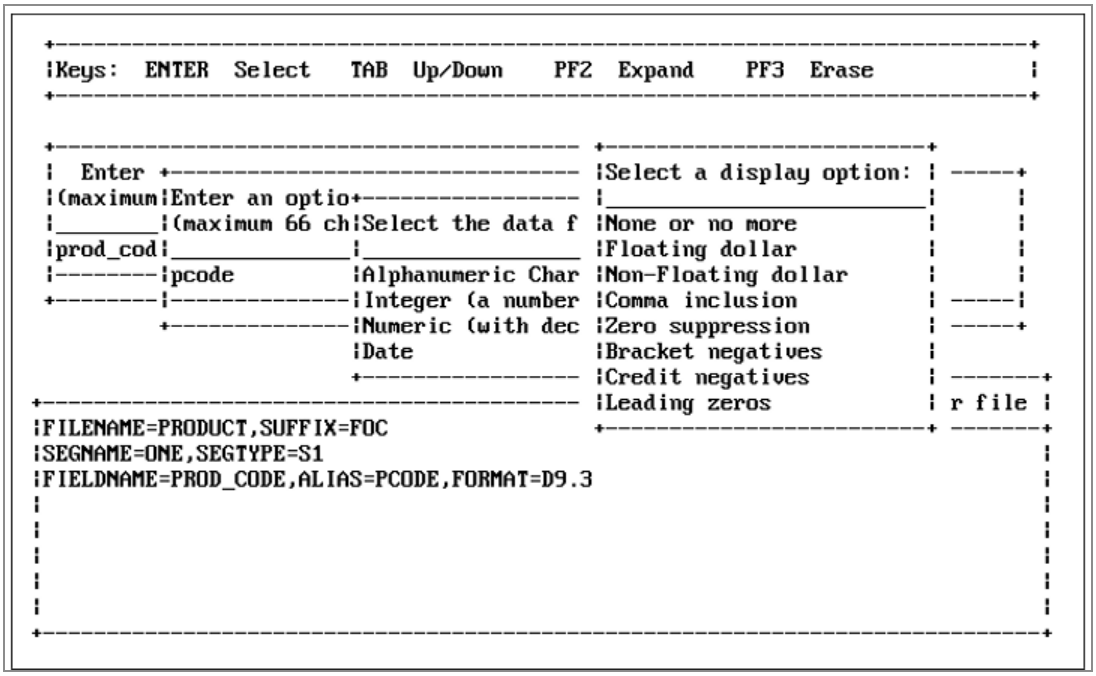


A numeric field consists of one to 15 digits including a decimal point. Numeric data may contain decimal places. If you select this option, FileTalk prompts you for the total field length (which includes the decimal point and decimal positions), the number of decimal places, and its display options, as shown in the following images:

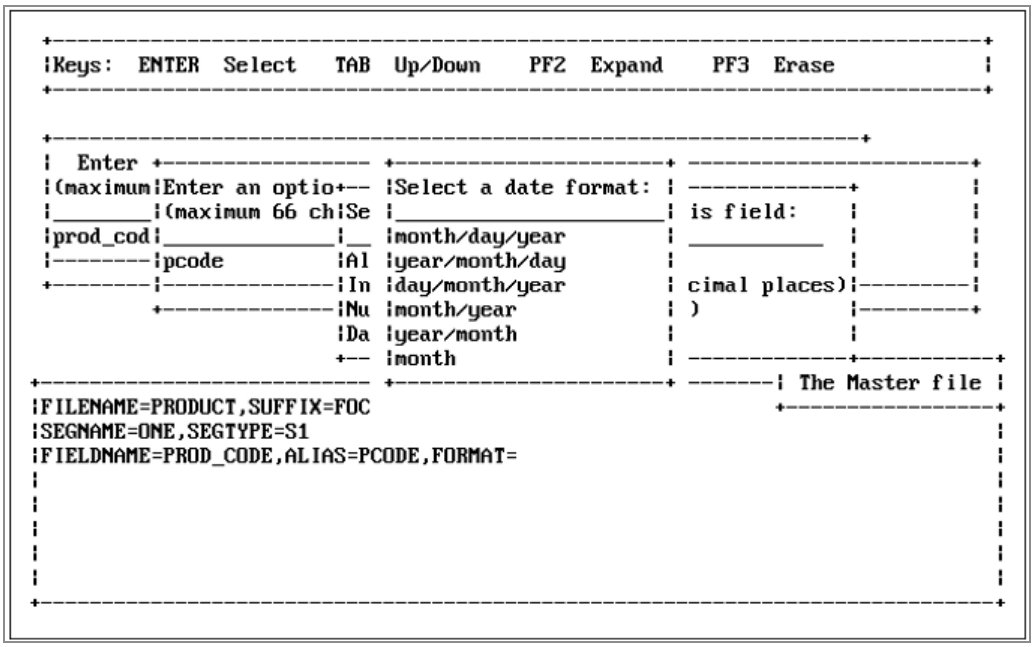




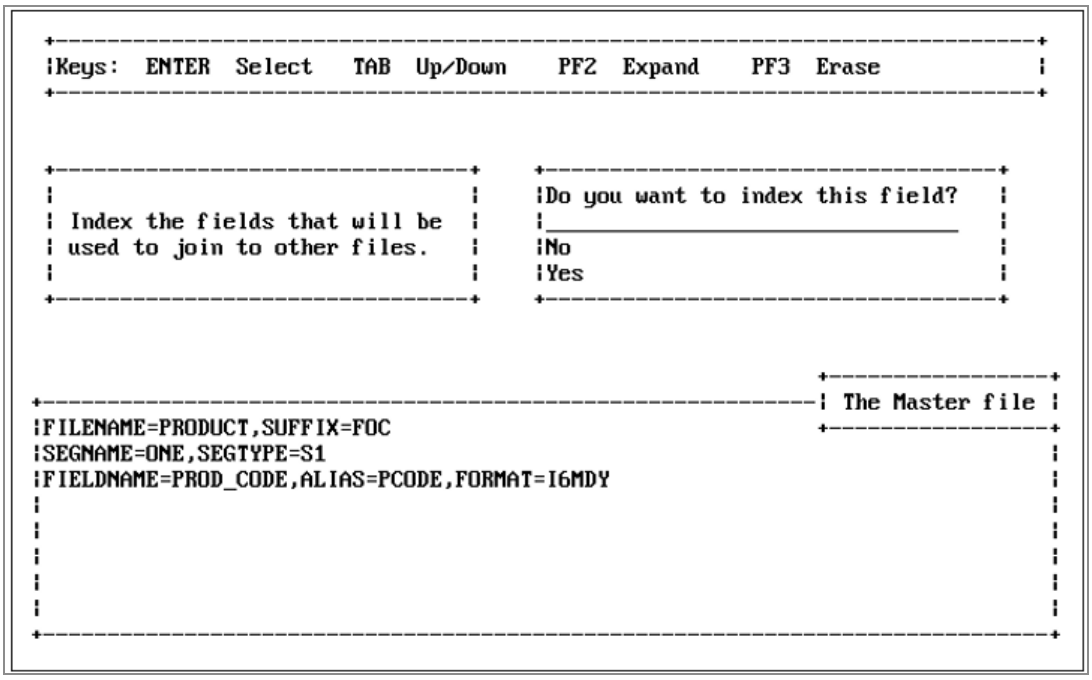
Some of the display options shown are mutually exclusive. For example, the *Floating dollar* and *Non-Floating dollar* options shown in the following image:



When you select the date format, the following format options display, as shown in the following image:



For example, select month/day/year and the following screen appears:



Defining Field Format for the Tutorial

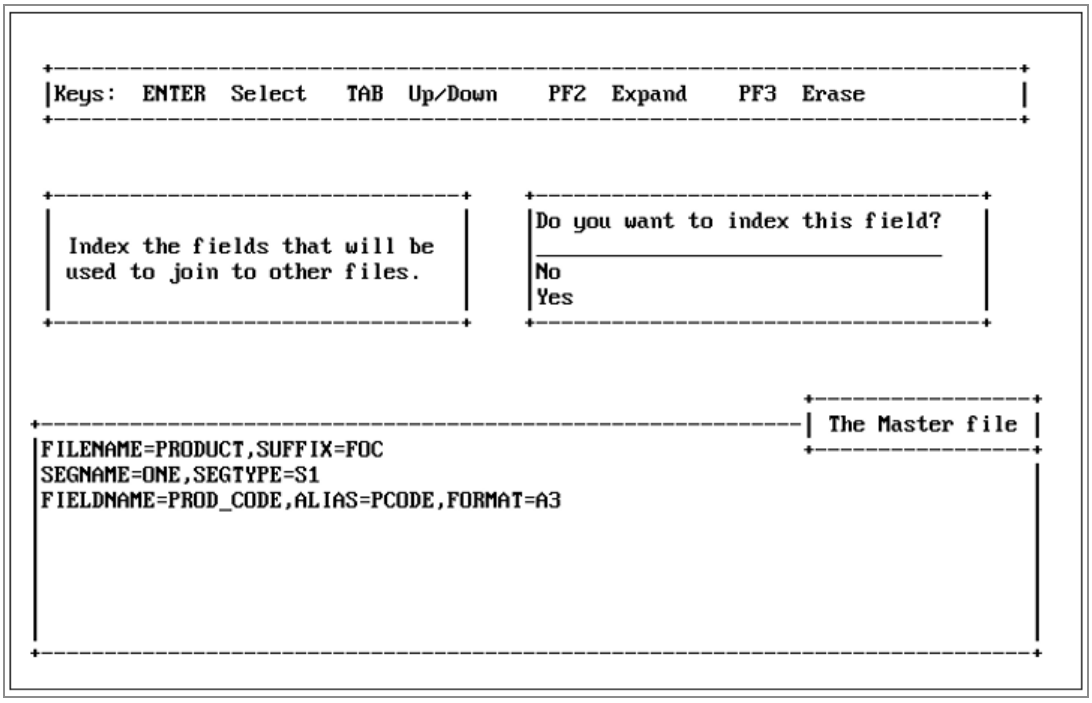
Now, continue with the file you are creating and for PROD_CODE, select the following data type:

```
Alphanumeric Characters
```

To specify the field size for this field, type:

```
3
```

PROD_CODE is now a three-character alphanumeric field, as displayed in the following image:



Indexing Fields

You can join files together to produce a single report if the files share common index fields. Indexed fields can also speed retrieval. In FOCUS syntax, only the TO field named in the JOIN command must be indexed. For information on joined databases (views), see Introducing Talk Technology.

For the Master File you are creating, make PROD_CODE an indexed field. Since you have entered the field size, FileTalk now prompts, *Do you want to index this field?* Move the cursor to the following option and press Enter:

```
Yes
```

Now that you have named the first field, indicated its format, and specified that it is an indexed field, the definition for PROD_CODE is complete.

Next, you add the remaining fields to this Master File. FileTalk prompts, *Any more fields?*. Select:

```
Yes
```

Add each of the following fields with their corresponding aliases and formats. Follow the same steps you followed for completing the description of PROD_CODE. Do not index any of these fields. Use this chart, proceeding from left to right, as a guide for your selections:

Fieldname	Alias	Format and Length	Index
PROD_NAME	ITEM	Alphanumeric, 15 characters	No
PACKAGE	SIZE	Alphanumeric, 12 characters	No
UNIT_COST	COST	Numeric (with decimal places), five digits, two decimal places, floating dollar sign (selected from the display options). After you select *Floating dollar*, select *None or no more* from the display options window in order to move on to the window for indexing fields.	No

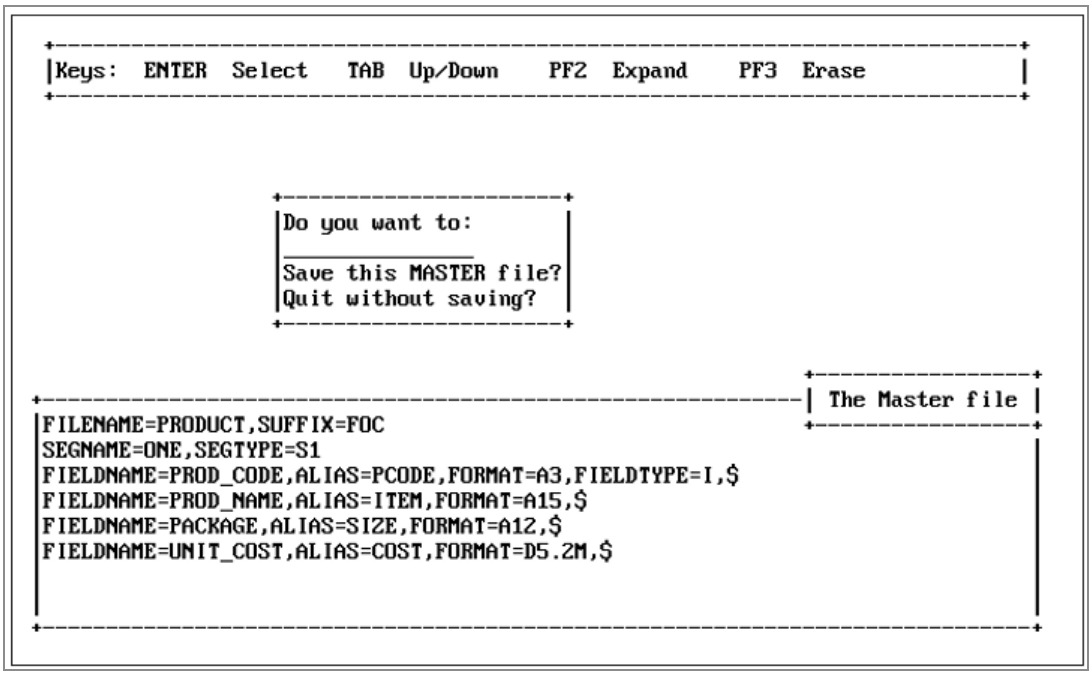
When you finish creating these three fields, FileTalk prompts you for *Any more fields?*. Select:

```
No
```

Now, with the file declared, the segment declared, and the fields declared in the PRODUCT Master File, the next step involves saving your work and ending your FileTalk session.

Saving the Master File

When you have defined all the segments and fields for your Master File, FileTalk displays the *Do you want to:* screen, as shown in the following image:



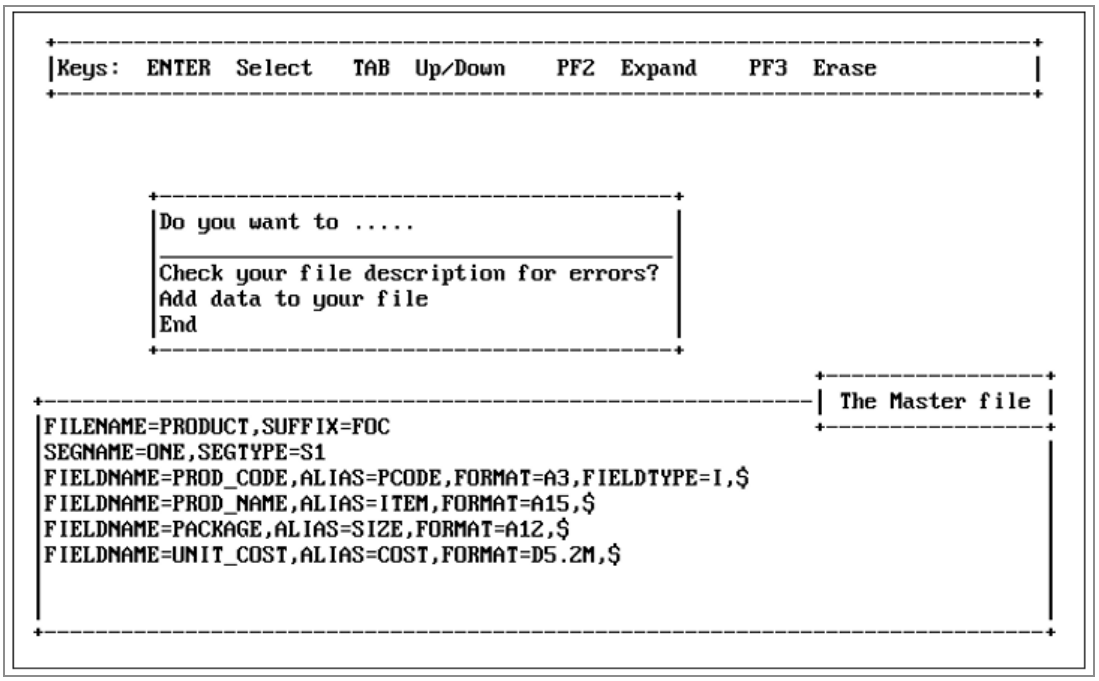
Save the Master File you have just created by selecting:

```
Save this MASTER file?
```

Under MVS, the description is placed in a partitioned dataset (PDS) allocated to the ddname MASTER. The membername is the filename you specified on the first FileTalk window. If the ddname MASTER is not allocated, FOCUS automatically allocates it as a temporary PDS. All members in this PDS that you wish to keep must be copied to a permanent PDS before you log off. If you allocate the ddname MASTER before entering FOCUS, FileTalk uses that allocation.

Checking Your Description

When you select the save option for your Master File, FileTalk displays options that enable you to check your Master File for errors, add data to your file, or end your FileTalk session, as shown in the following image:



These options are explained in the following table:

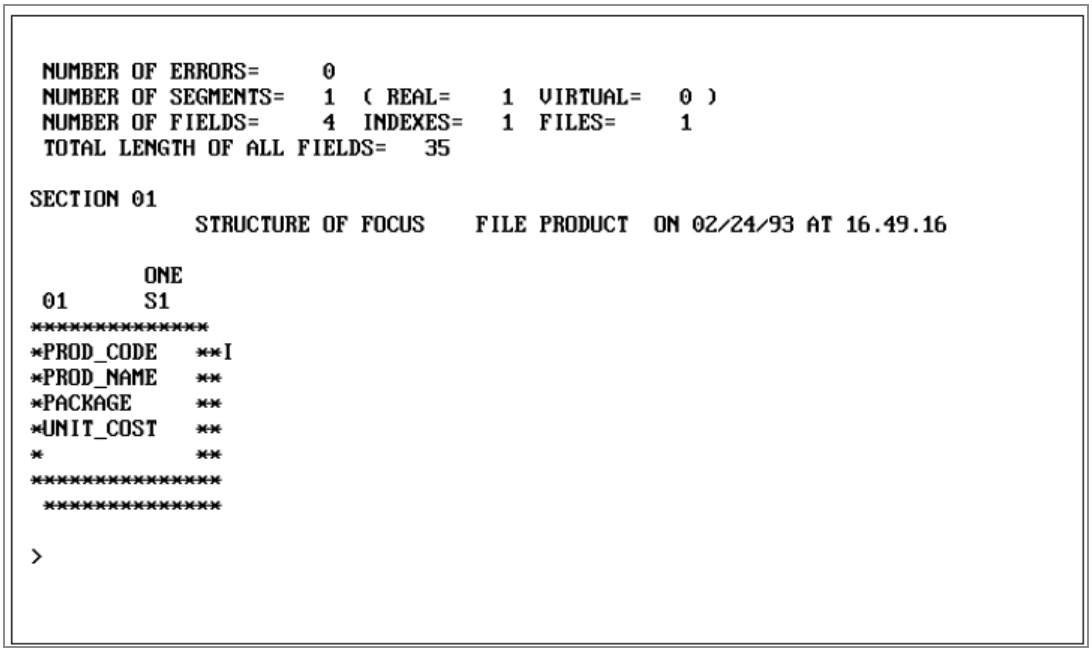
Option	Function
`Check your file description for errors?`	FOCUS saves the Master File, displays a picture of the file, and returns you to the FOCUS command prompt (as you will see when you actually select this option later in this section).
`Add data to your file`	FileTalk automatically invokes the Automatic Data Management System, AUTOMOD. You can use this system from within FileTalk or from the FOCUS prompt to add, delete or change data in new or existing FOCUS files. The AUTOMOD facility is intended only for single-segment files. To add data to multi-segment files, use the ModifyTalk facility. For more information on ModifyTalk, see Saving the Master File. To invoke AUTOMOD outside of FileTalk, at the FOCUS command prompt type the following command: `EX AUTOMOD`

Option	Function
END	FileTalk saves the Master File and returns you to the FOCUS command prompt. Under MVS, the Master File becomes a member of the partitioned dataset (PDS) which has the ddname MASTER. The membername is the filename you specified on the first FileTalk window. If the ddname MASTER is not allocated, FOCUS automatically allocates it as a temporary PDS. All members in this PDS that you wish to keep must be copied to a permanent PDS before you log off. If you allocate the ddname MASTER before entering FOCUS, FileTalk uses that allocation.

Now, for the Master File you have created, select the following option:

```
Check your file description for errors?
```

Notice that statistics about the file, as well as a diagram of the segments, keys, indexes, and fields display. Each box in this diagram corresponds to a segment in the Master File, and these boxes list up to four fields from each segment. Also, notice that when you check your file description for errors, you are returned to the FOCUS command prompt (>). This means your FileTalk session is over and you are no longer in FileTalk.



ModifyTalk

This topic describes how to use ModifyTalk to create full-featured MODIFY FOCEXECs for FOCUS databases that accept and display data using CRTFORM screens. You may use ModifyTalk to generate procedures (MODIFY FOCEXECs) for FOCUS databases which consist of single or multiple segments. ModifyTalk prompts you to select options from a series of FOCEXEC Processing Windows to describe processing logic, then it generates the procedure. By working through the **tutorials** later in the chapter, you learn how to create default and customized procedures.

ModifyTalk Features

ModifyTalk automates the process of creating a MODIFY procedure, with windows that list options to help you:

- Create the procedure you want to generate.
- Specify the segments and fields you want to see at each step of the MODIFY procedure.
- Select the MATCH and NOMATCH actions you want to take.

You can use ModifyTalk to generate default or customized procedures:

- Default procedures can perform all MODIFY actions except deleting data in the segments you specified. However, you can delete information if you choose to use multiple record screens **(See Tutorial: Generating a Default Procedure on page 5-10.)**.
- Customized procedures can perform MODIFY actions you specify for each segment.

Furthermore, the segments you use in the MODIFY procedure are called *active* segments. You do not need to select all the segments in a path if you only want to modify a descendant segment. ModifyTalk automatically generates the commands needed to reach that segment.

The FOCEXECs generated by this facility are extensively commented because ModifyTalk is self-documenting. During your ModifyTalk session, your selections are copied both to the

bottom window and to the top of the MODIFY procedure. The generated FOCEXECs use Case Logic.

Note:

- When the FOCEXECs created with ModifyTalk are executed, they use special function keys. These keys are listed in Function Keys for FOCEXECs Created With ModifyTalk for FOCEXECs Created With ModifyTalk.
- ModifyTalk does not create procedures that read incoming data using the FIXFORM, FREEFORM, or PROMPT statements.
- Cases and variables used by ModifyTalk are listed in ModifyTalk FOCEXEC Cases and Variables.
- A sample FOCEXEC generated by ModifyTalk is supplied in ModifyTalk-Generated FOCEXEC.

MODIFY Facility

Data files are rarely static. The information they store can change constantly. To maintain your files, you need to be able to view, update, add, and delete records. These functions are the basis of a data maintenance procedure.

In FOCUS, these functions are performed by the MODIFY facility. For more information on MODIFY, see the *FOCUS for IBM Mainframe User's Manual.*

ModifyTalk Requirements and the Role of ibi FOCUS File Structures

Before you can generate a MODIFY procedure with ModifyTalk, you must have:

- A Master File for the data file you want to use. If you do not have a Master File, you can create one with FileTalk, with TED, or with your system editor.
- A data file. (The examples in this chapter use the SALES and EMPLOYEE databases. If you have not already done so, create the required databases as described in Introducing Talk Technology.)

- Knowledge of how the data file is structured. For example, you would need to know the names of the fields you want to modify, the names of the segments that contain these fields, and the segments in the path to the active segment you are using for this procedure.

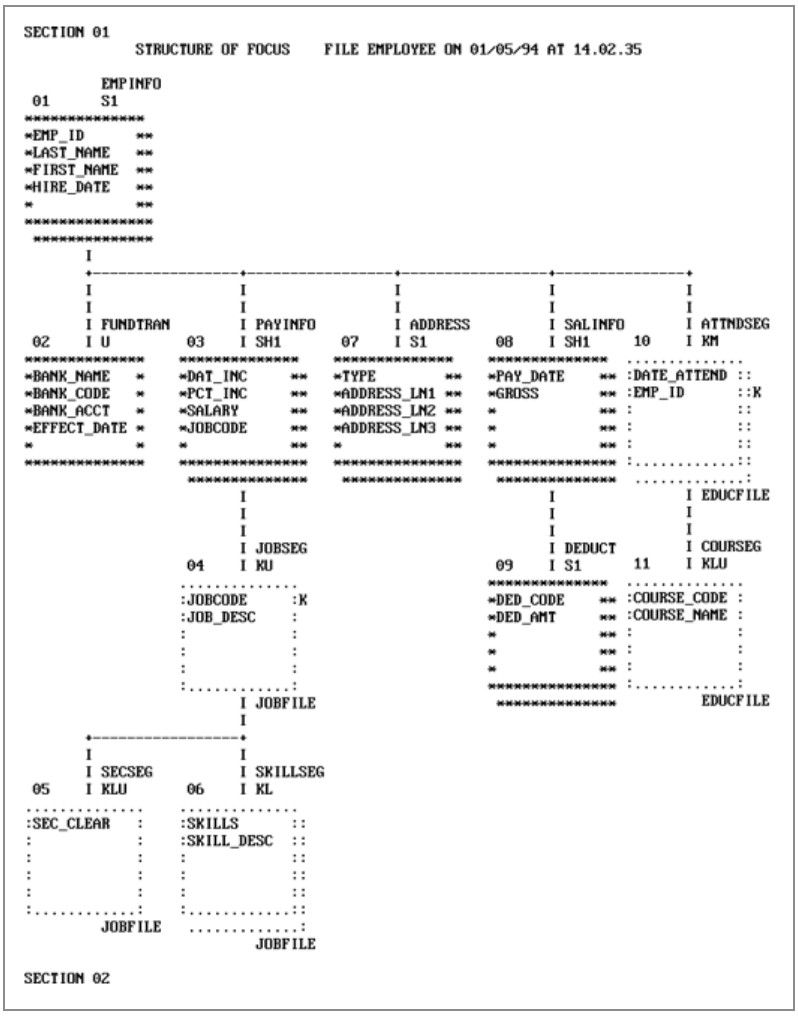
You can produce a structure diagram of your Master File by issuing at the FOCUS command prompt the FOCUS CHECK FILE command. Type the following command and press Enter:

```
CHECK FILE filename PICTURE
```

Where *filename* is the name of your Master File. For example, if you type the following command:

```
CHECK FILE EMPLOYEE PICTURE
```

FOCUS automatically displays the structure diagram of the EMPLOYEE Master File, as shown in the following image:



Note that the CHECK FILE command first shows statistics about your file (not shown), then displays a diagram of all the segments, keys, indexes, and fields.

Each box on this diagram corresponds to a segment in the Master File, and these boxes list up to four fields from each segment. To see all fields contained in a segment, refer to the file's Master File. The Master Files for the EMPLOYEE and SALES databases appear in **Appendix A, Overview of the Master File Description.**

When you finish selecting the segments you want to modify, the CHECK FILE diagram can help you plan the processing logic for your procedure. **The examples in Tutorial: Two Sample FOCEXECs on page 5-36** illustrate how the CHECK FILE command is used to plan a customized FOCEXEC. For more information about FOCUS files, see the *FOCUS for IBM Mainframe User's Manual*.

Role of ibi FOCUS Key Fields

Each segment in a data file has its own key field or fields. FOCUS uses these keys to sort and identify the individual entries. For example, individual products in an inventory file might be identified by a unique product number stored in a field called PROD_CODE.

Before you can modify the instances in a segment, you must first supply all the key field values needed to retrieve a record. Key field values can determine if a particular value exists in the segment. The procedures generated by ModifyTalk automatically prompt you for key field values on Key Values screens for a particular segment.

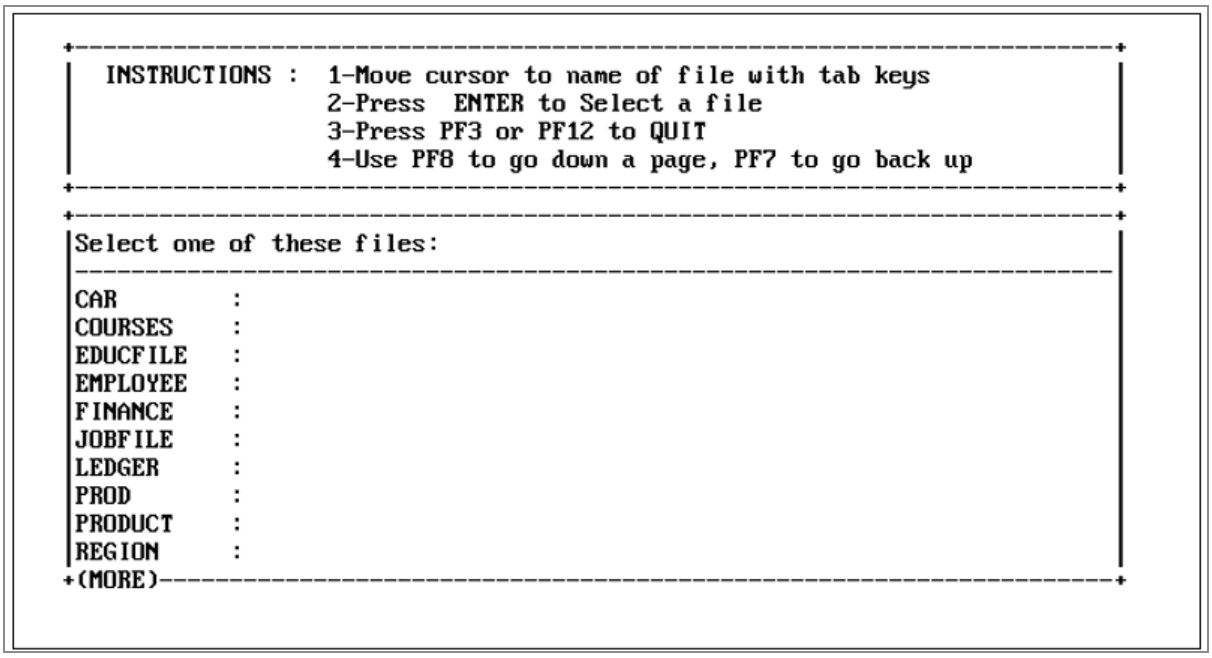
After you supply a value for the key field, the MODIFY procedure determines if there is an instance with a key field that matches your input. This is specified with the FOCUS MATCH or NOMATCH actions you select during the ModifyTalk session.

Entering ModifyTalk and Selecting a File

This section describes how to enter ModifyTalk and displays the first screen you see in a session. Do not do this until or unless you have already created the required databases, described in Introducing Talk Technology. To invoke ModifyTalk, at the FOCUS command prompt type the following command and press Enter:

```
MODIFYTALK
```

The first screen displays a list of files available for modification as shown in the following image:



To select a file, simply position the cursor on the file of your choice and press Enter.

Alternate Ways to Enter ModifyTalk

You can also invoke ModifyTalk by typing the following command at the FOCUS command prompt:

```
MODIFYTALK FILE filename
```

where *filename* is the name of the file you want to use. ModifyTalk automatically skips the File Selection window.

Other ways to enter ModifyTalk include selecting it from the FOCUS Menu facility or the FOCUS Toolkit facility. Instructions for using both facilities are provided in Introducing Talk Technology.

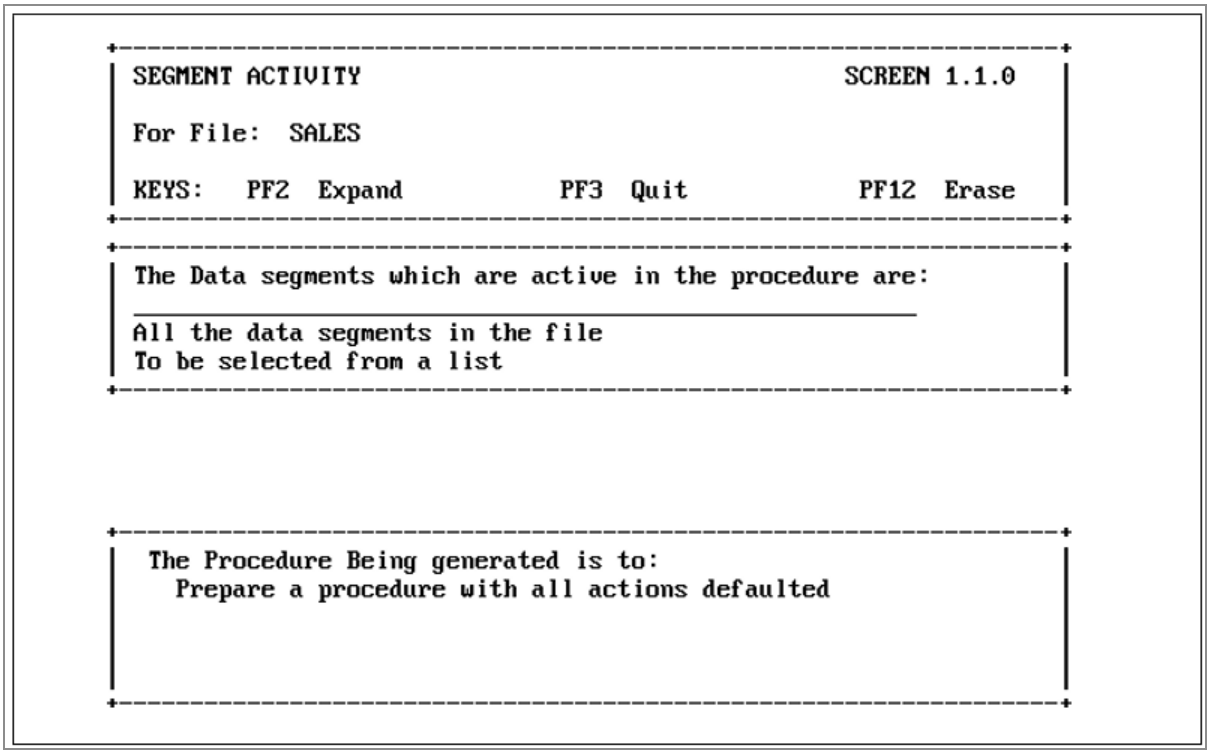
ModifyTalk Window Layout

ModifyTalk screens are divided into three main areas:

- The top window, which displays instructions and processing information.

- The middle windows, which display selections.
- The bottom window, which displays your current MODIFY procedure description.

The following image illustrates the three windows:



Top Window

The top window displays several items of information that indicate the current status of your ModifyTalk session:

`STATUS MESSAGES`	Display in the upper left corner of the window. They identify the part of the MODIFY process you are describing.
`SCREEN NUMBERS`	Display in the upper right corner of the window. They help you track your current location in ModifyTalk.

FILE AND SEGMENT IDEN- TIFIERS	Display in the middle of the window> They are messages that remind you where you are in the data file.

Middle Windows

Selection windows appear in the middle of a ModifyTalk screen. These windows list options that enable you to determine the actions in the procedure you are creating. The windows across the top are called FOCEXEC Processing windows, and the windows on the right are called Segment Activity windows.

Your selections from FOCEXEC Processing windows determine how the MODIFY FOCEXEC is constructed and stored. FOCEXEC Processing windows include the following:

Process Selection Window	Enables you to specify a default or customized MODIFY procedure
Segment Activity Window	Enables you to select the active segments in the procedure.
Multiple Record Selection Menu Window	Enables you to select multiple record processing for the procedure.
End of Generation Selec- tion Menu Window	Enables you to specify whether to save the generated procedure or to execute it without saving it.
Saved Procedure Selec- tion Menu Window	Enables you to specify how to process the saved procedure.

In addition, your selections from the Segment Activity windows determine functions of your ModifyTalk FOCEXEC. These windows are:

`Matching Data Activity Selection Window`	Enables you to select an action when an instance matches the key field you specified.
`No Matching Data Activity Selection Window`	Enables you to select an action when a matching instance for the key field is not found.
`Control Flow Window`	Enables you to specify the next segment to process after Matching or No Matching Data Activity Selection window selections.

Note: Segment Activity windows enable you to select options for each active segment in the MODIFY procedure. You can determine the processing order for segments in a file by selecting options from the Control Flow windows.

Bottom Window

Your choices from the selection windows are automatically displayed in the bottom window of your screen. These selections comprise a running description of the MODIFY procedure.

If this description becomes too long for the bottom window you can review its entire contents by pressing PF2. This places your description in Expanded Mode. You can scroll backward in the display using the PF7 key and forward using the PF8 key. When you are finished viewing the description, press PF2 again to resume your ModifyTalk session.

Unlike other Talk Technology products, your selections in ModifyTalk do not immediately create FOCUS commands. Instead, they describe the physical characteristics and processing logic of the procedure you are creating. ModifyTalk only generates the FOCUS syntax when your description is complete or when you reach the end of your ModifyTalk session.

When the description of your MODIFY procedure is complete, select the SAVE option to generate the MODIFY FOCEXEC. The description is automatically incorporated as comment

lines in the procedure. Therefore, the description not only determines the contents of the procedure, but also serves as comments in the MODIFY FOCEXEC.

Tutorial: Generating a Default Procedure

You can generate a comprehensive MODIFY FOCEXEC simply by accepting default ModifyTalk options. When you choose to generate a default procedure, you make choices that determine the physical characteristics of the resulting FOCEXEC. ModifyTalk provides the processing logic.

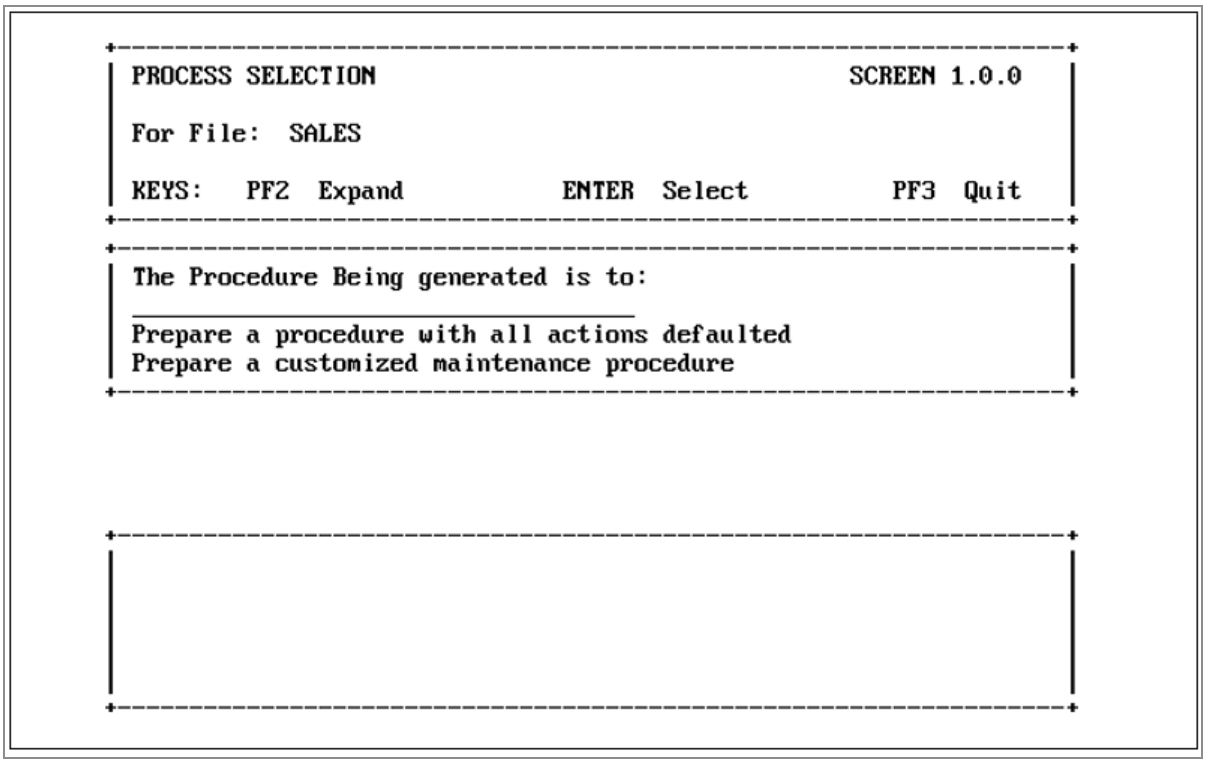
When you finish making your choices, ModifyTalk generates a MODIFY FOCEXEC that offers view, update, and new data input choices for each of the segments selected. You can also delete segment instances if you choose to use multiple-record screens. The FOCEXEC processes the active segments top-down and left-to-right in their logical order as displayed by the CHECK FILE command.

Since default procedures produced in ModifyTalk provide all MODIFY capabilities for each segment of the file, you do not need to plan the procedure's processing logic. You simply select options from five FOCEXEC Processing windows.

Start this ModifyTalk tutorial by following the steps described in Entering ModifyTalk and Selecting a File. You will see a list of the databases available to you. Position the cursor next to SALES and press Enter.

Process Selection Window

The first window you see is the *Process Selection* window. ModifyTalk prompts you for a selection from this window, as shown in the following image:

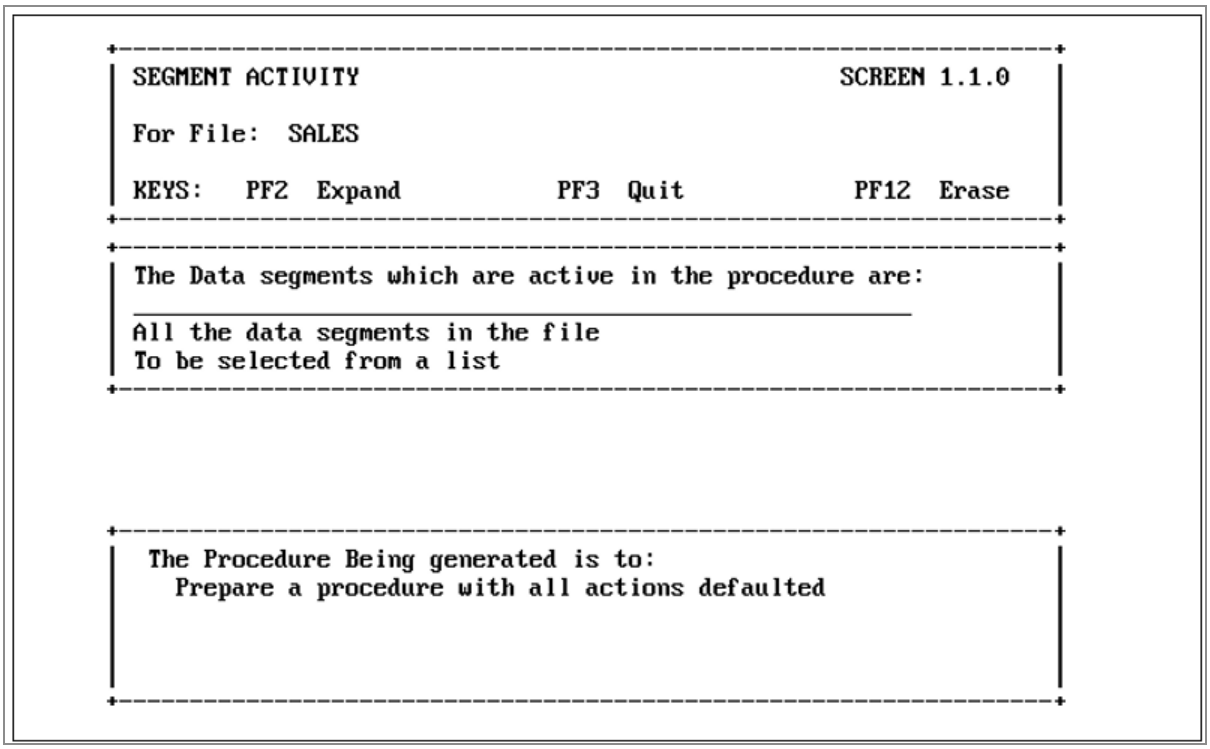


Choose the following option and press Enter:

```
Prepare a procedure with all actions defaulted
```

Segment Activity Window

Next, the Segment Activity window prompts you for the active segments in your procedure, that is, the segments you want to use in the MODIFY FOCEXEC, as shown in the following image:



Select the following option and press Enter:

```
All the data segments in the file
```

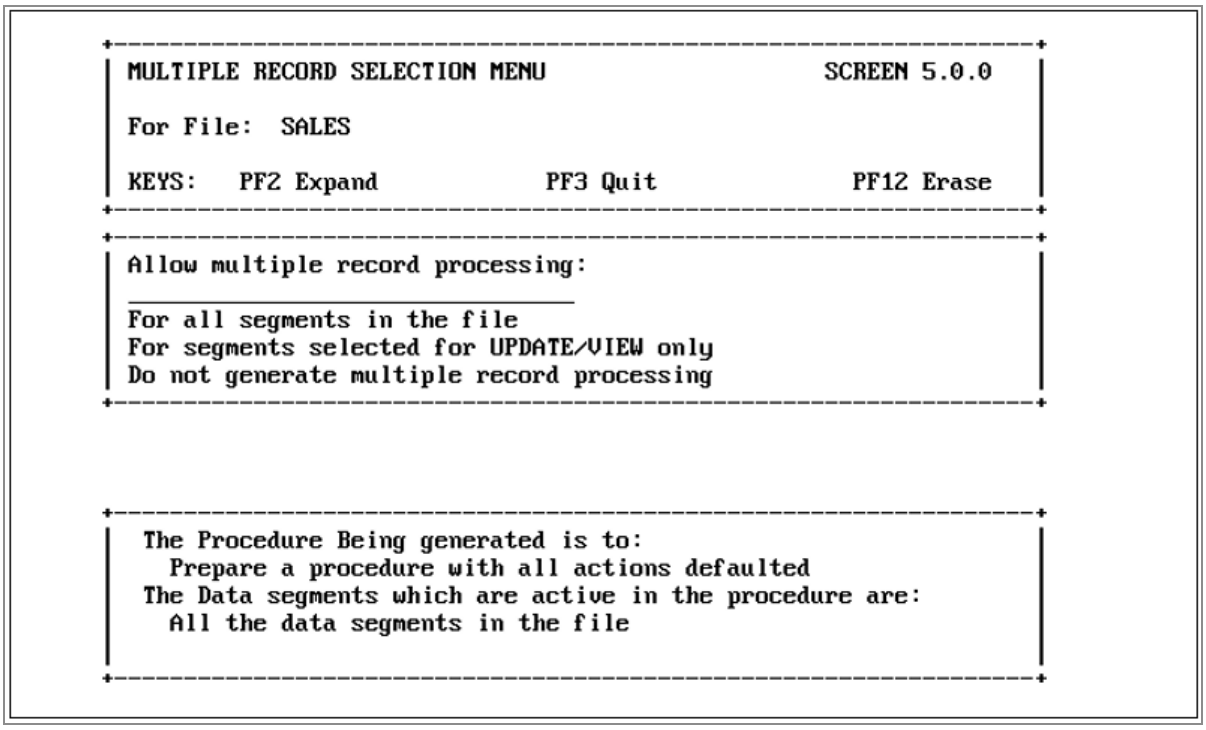
If you choose the option *To be selected from a list*, you are shown a window listing the names of all the available segments. Select these segments by moving the cursor to your first selection and pressing PF9. Then, move the cursor to your second selection and press PF9 again. Repeat these steps until you complete your last selection. Press Enter to add all your choices to the MODIFY description.

If you want to use only one segment from the list, simply move the cursor to that name and press Enter.

You next indicate whether multiple-record processing will be available.

Multiple Record Selection Window

The Multiple Record Selection Menu window enables you to specify whether or not multiple record processing is available for the active segments in your procedure, as shown in the following image:



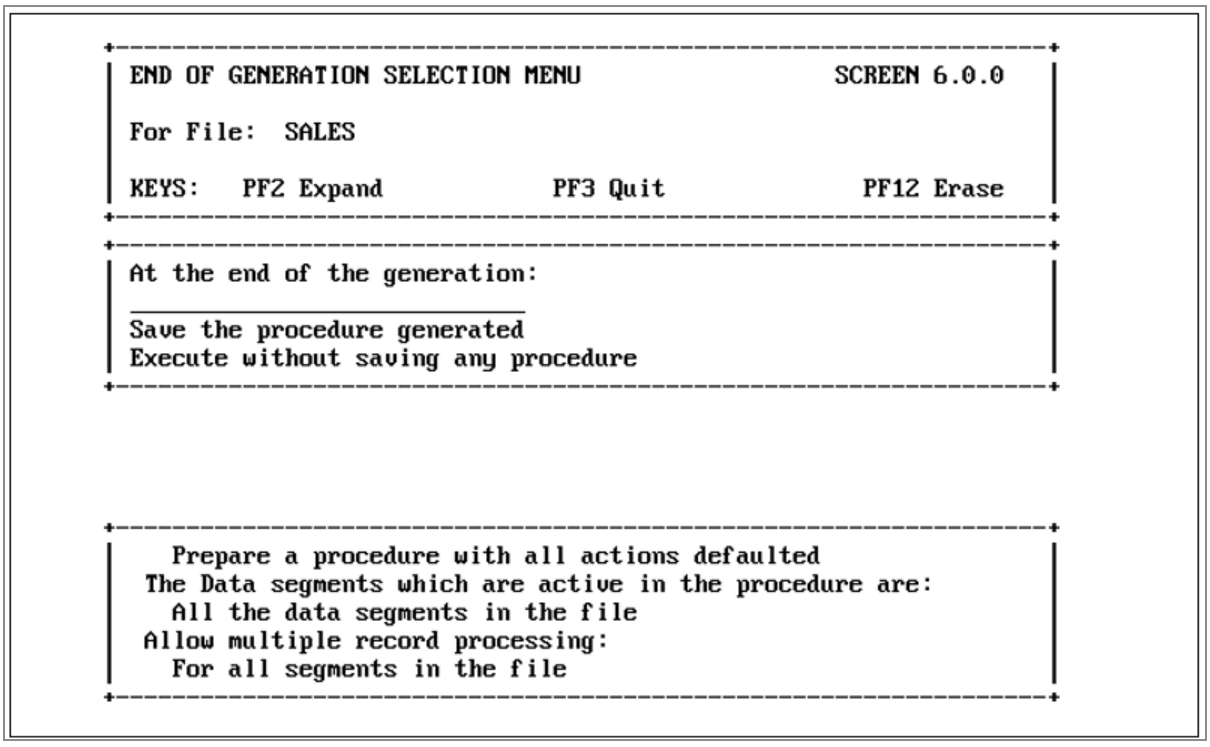
Because the user can delete instances from segments in multiple record screens, you should be careful about assigning this capability. For this example, this capability is provided. Therefore, select the following option and press Enter:

```
For all segments in the file
```

End of Generation Selection Menu Window

The End of Generation Selection Menu window immediately follows the Multiple Record Selection window. It displays the following options, as shown in the following image:

- Save the procedure generated
- Execute without saving any procedure



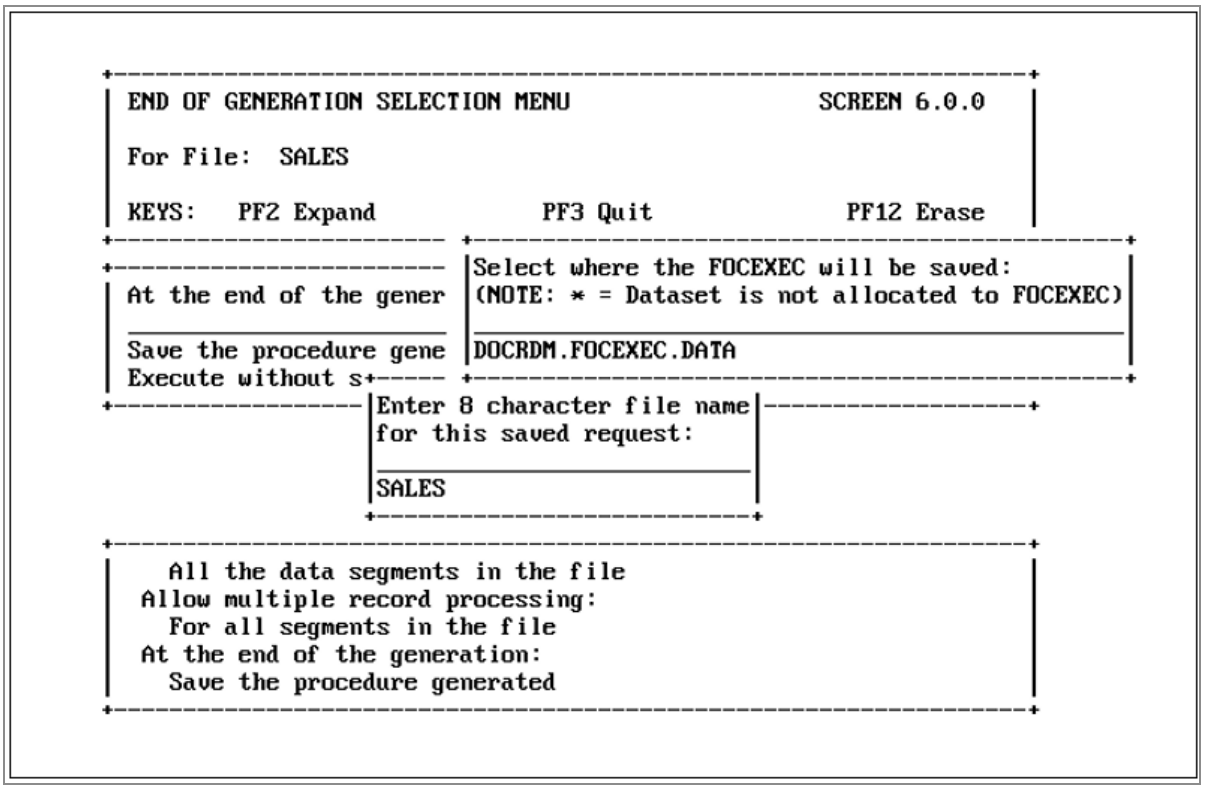
Select the following option and press Enter:

```
Save the procedure generated
```

ModifyTalk prompts you to *Enter 8 character file name for this saved request*. Type the following name and press Enter:

```
Sales
```

- Under MVS, when you save a file, ModifyTalk displays the following additional screen in which you select the partitioned dataset, as shown in the following image:

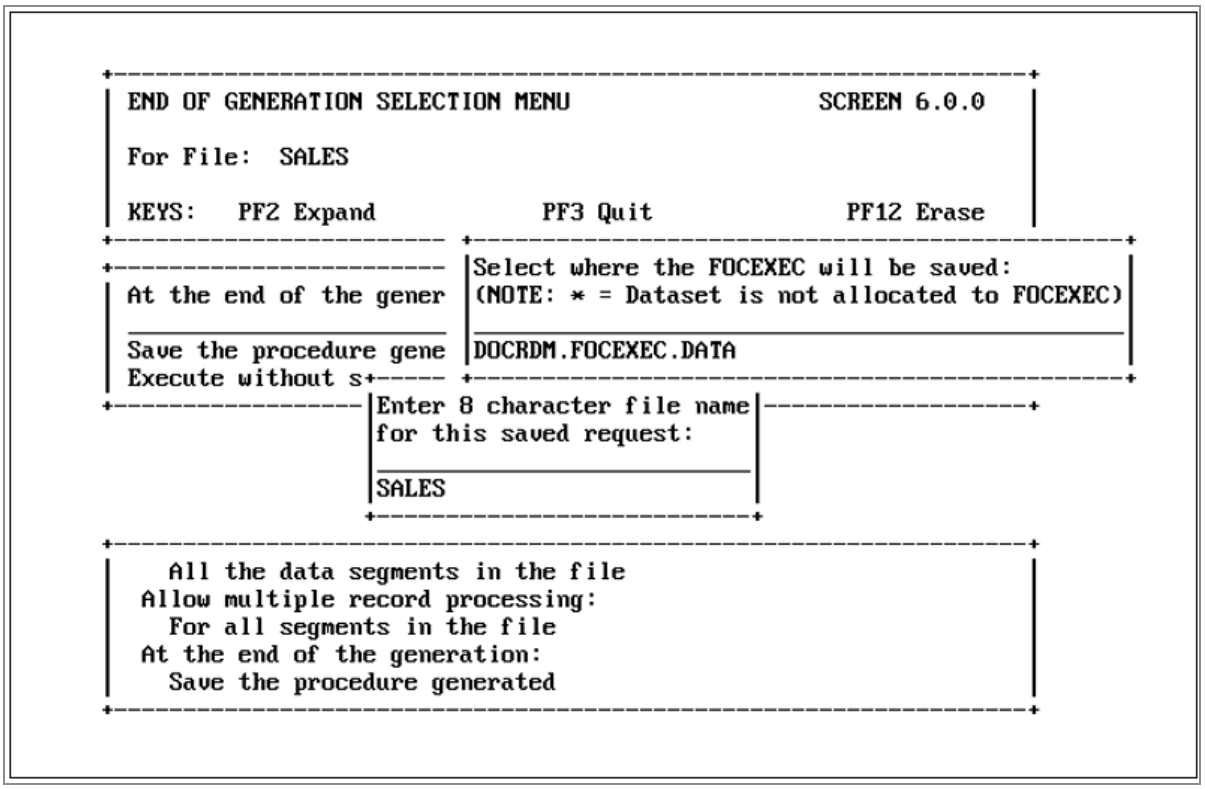


Once you provide a membername, ModifyTalk displays a list of partitioned datasets allocated to ddname FOCEXEC where DISP=OLD or NEW. You can indicate the dataset to which the procedure should be saved.

If no such partitioned datasets are available, ModifyTalk looks for the dataset *prefix.FOCEXEC.DATA*, where *prefix* is either your MVS user ID, or the user ID set by the FOCUS SET PREFIX command. See the *FOCUS for IBM Mainframe User's Manual* for information on this command.

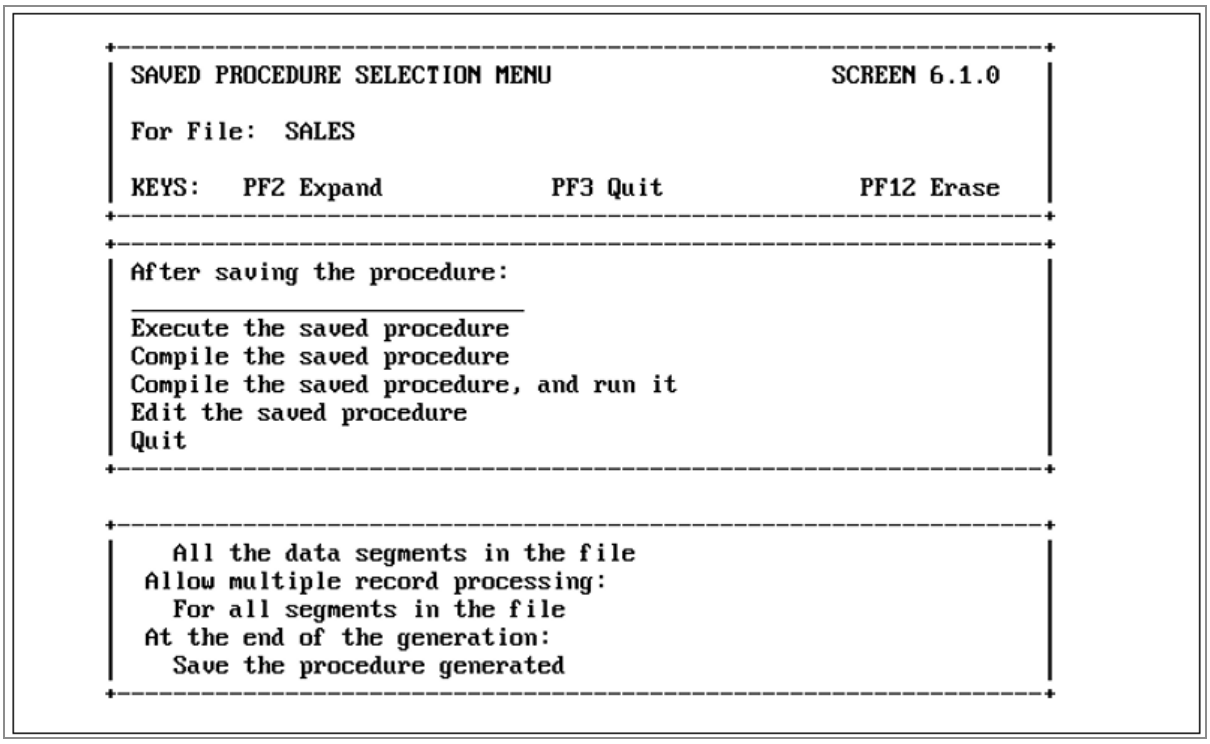
If none of these partitioned datasets is available, ModifyTalk writes the procedure to a temporary sequential file named MODTALK. This file is erased at the end of your session, unless you take the appropriate steps to save it.

If you specify a pre-existing membername in a partitioned dataset, you are prompted to supply a new name or overwrite the existing member. Existing sequential datasets are overwritten. If you attempt to save a member to a partitioned dataset for which you do not have write access, then the following window appears, and you must supply a new name, as shown in the following image.



Saved Procedure Selection Menu Window

Next, you indicate how to use the saved procedure, as shown in the following image.



Your options are:

Option	Description
`Execute the saved procedure`	Immediately execute the saved MODIFY procedure.
`Compile the saved procedure`	Compile the saved procedure.
`Compile the saved procedure, and run it`	Compile the saved procedure and immediately execute it.

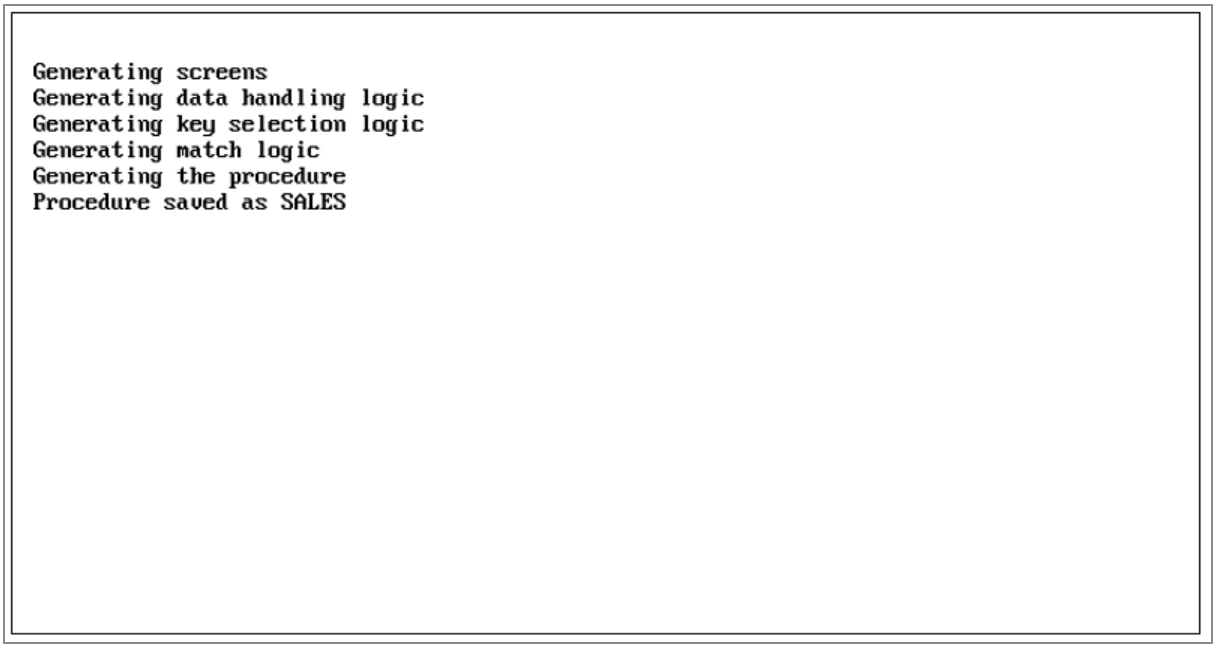
Option	Description
`Edit the saved procedure`	Load the saved MODIFY procedure into the TED editor. When you exit TED you are returned to the FOCUS command prompt. For further information about TED, the FOCUS editor, see the *FOCUS for IBM Mainframe User's Manual*.
`QUIT`	Return to the FOCUS command prompt. The MODIFY procedure is stored in the named file.

Once your MODIFY procedure is working to your satisfaction, you can compile it. Compiling the procedure results in quicker, more efficient execution.

For this example, select the following option and press Enter:

```
Execute the saved procedure
```

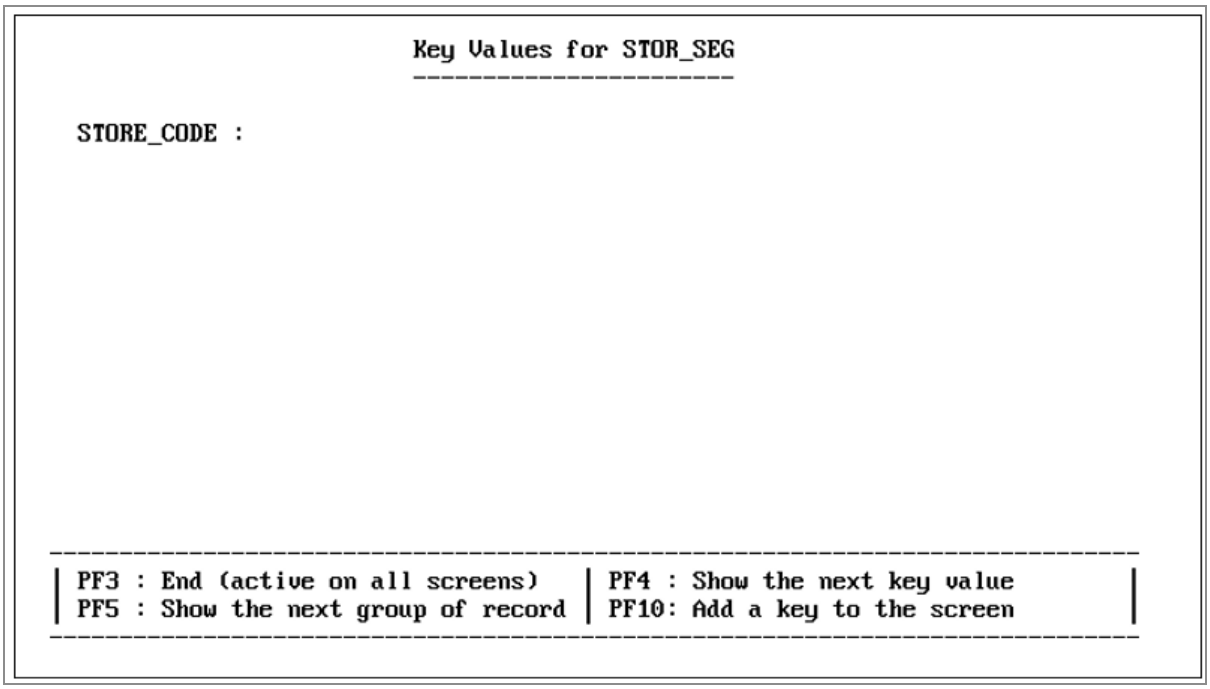
ModifyTalk automatically generates the MODIFY FOCEXEC. As it runs, ModifyTalk displays a series of messages describing each step of this process, as shown in the following image:



After the FOCEXEC is generated, it is immediately executed.

Executing the Default FOCEXEC

When the default MODIFY FOCEXEC you just created is executed, it next prompts you for the key values for the first segment (STOR_SEG) in the SALES file, as shown in the following image:

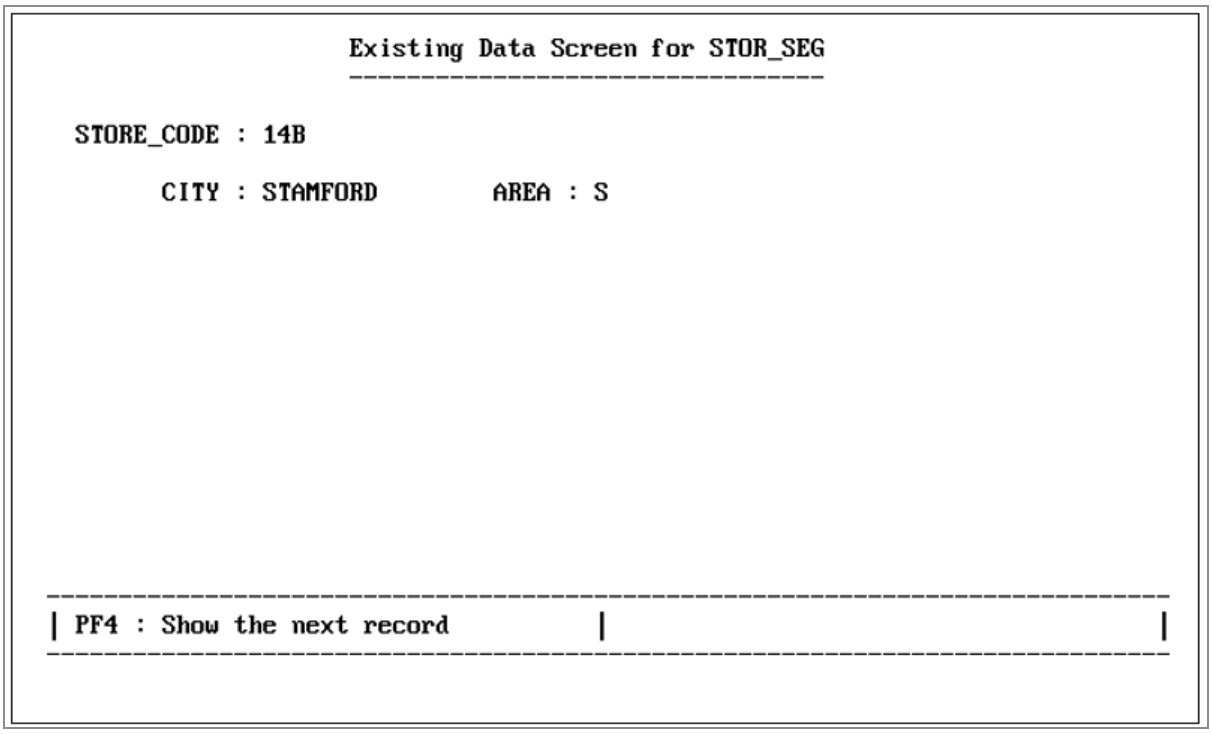


Retrieving a Segment Instance

When the procedure prompts you for a key value, there are two ways for you to retrieve a segment instance. You can enter an existing value in the input field. In this case, the input field is STOR_CODE. For example, type the following value and press Enter:

```
14B
```

Note that all alphabetic characters are displayed in uppercase. The procedure automatically displays the corresponding CITY and AREA values for STORE_CODE 14B on your screen. This screen is called the Existing Data Screen for STOR_SEG, as shown in the following image:



You can also retrieve a segment instance by pressing PF4 when ModifyTalk prompts you for a key field. Each time you press PF4, you see the next instance on the segment. When you reach the last instance, the procedure displays the following message, *End of file reached...Restarting at beginning*, and returns to the first instance of the segment.

This key can be used with either the Key Value Screen or the Existing Data Screen, in which case you see a complete segment instance each time you press the key.

For now, press Enter to continue.

Updating Fields on a Segment Instance

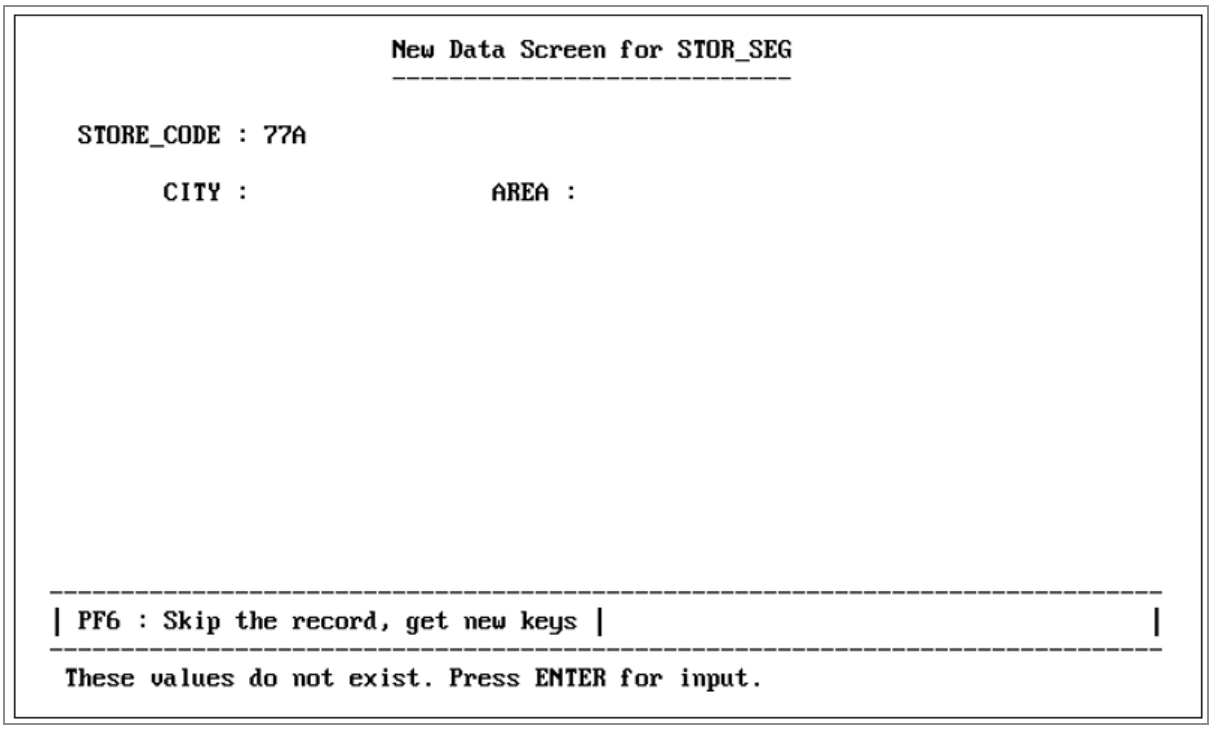
You can update the values of the non-key fields. Note that when you retrieve an instance the cursor is on the first non-key field. Use the Tab key to move to the fields you want to change. When you complete all your revisions, press Enter to write the information back to the database.

Adding New Instances

You can add new instances to the segment by entering a new key field value on the Key Values Screen. Since you are at the Existing Data Screen, press Enter to return to the Key Values Screen, and type the following value in the STORE_CODE key field:

```
77A
```

The procedure automatically determines that 77A is not an existing store code in the SALES segment and displays a New Data Screen for STOR_SEG. This screen prompts for new CITY and AREA field values, as shown in the following image:



Type the following value in the CITY field:

```
plattsville
```

Press Tab and type the following value in the AREA field and press Enter:

```
r
```

This new segment instance is now entered into the database. You now see the Key Value Screen for the DATE segment. When you add an instance to a segment, you get the Key

Value Screen for the next active segment so that you can continue to add data. To return to the Key Value Screen for STOR_SEG, press PF9, *Drop a key from the screen*.

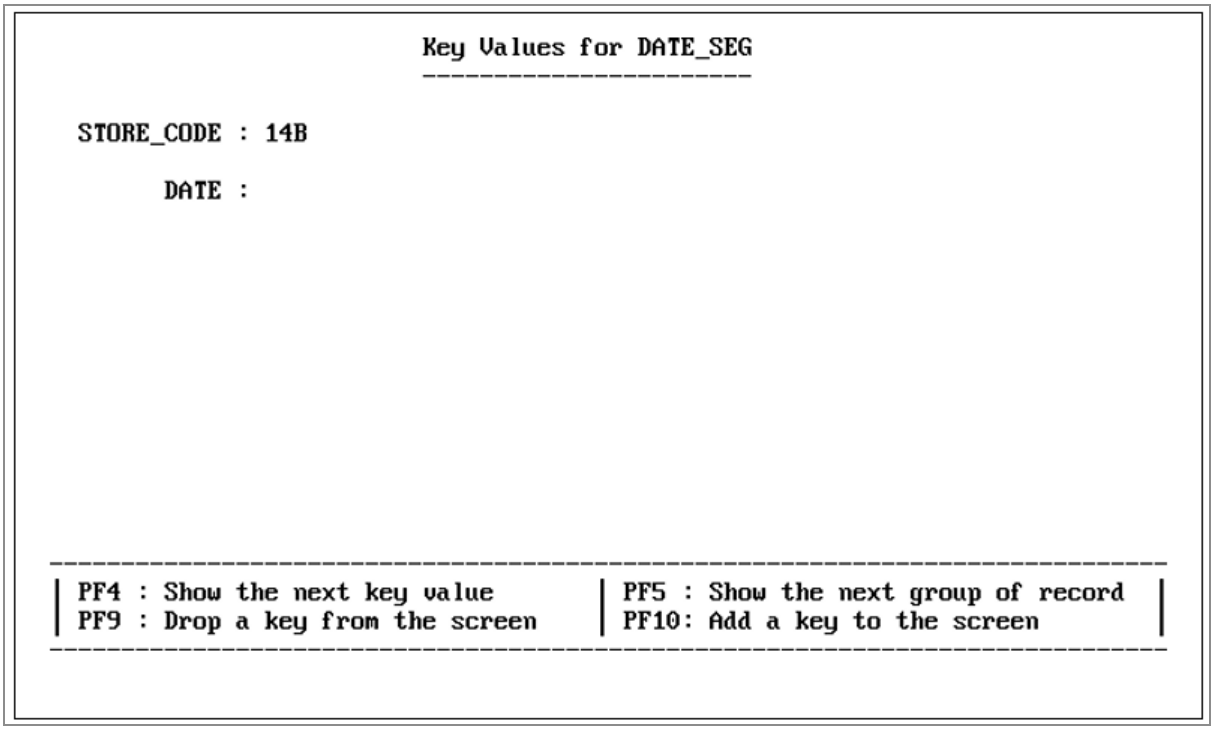
If you typed a new key field value without intending to add a new instance to the file, you can cancel the data input operation by pressing PF6.

Continuing to Descendant Segments

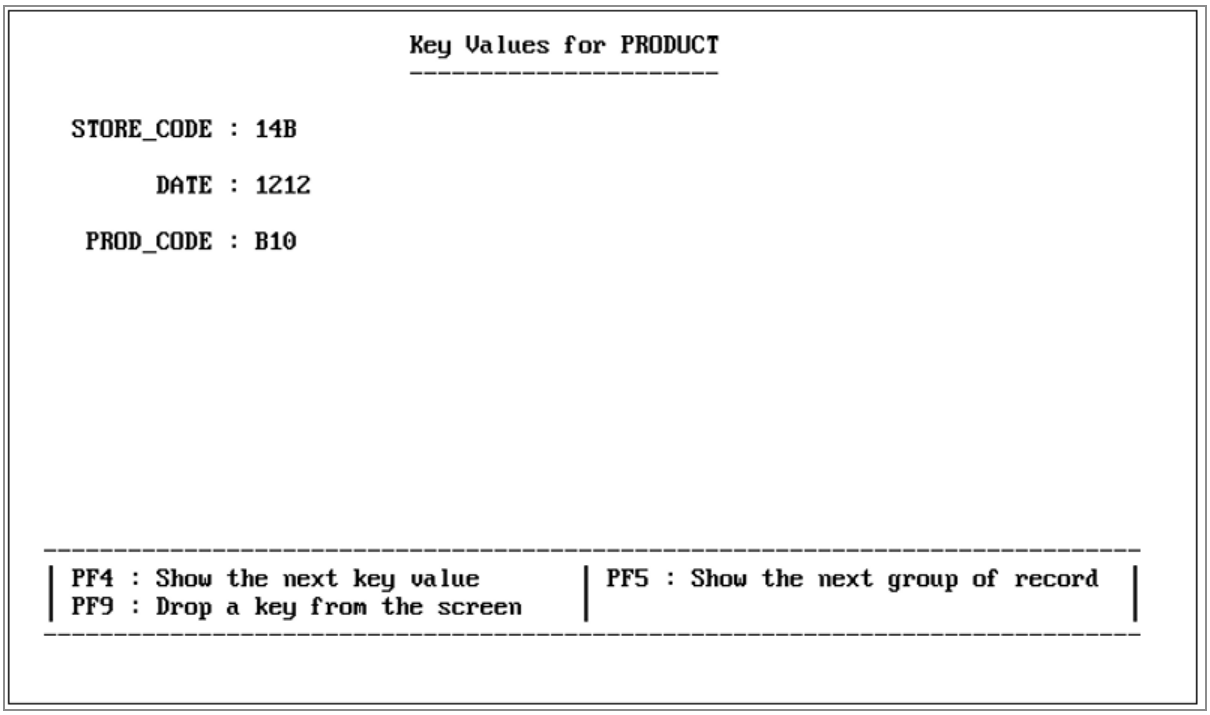
To continue from a segment instance to a descendant segment, press PF10, *Add a key to the screen*. Now, on the Key Values Screen for the STOR_SEG segment, type the following value for STORE_CODE and press F10:

```
14B
```

The SALES file is a multi-segment file, and you press PF10 to display the first field in the descendant segment. Your procedure displays the Key Values Screen for the DATE_SEG segment, as shown in the following image:



Press PF4 to display the first instance in the DATE_SEG segment. The first sales date is shown. Now, press PF10 to continue to the PRODUCT segment and press PF4 to see what products were sold on this date, as shown in the following image:



You see the Key Value Screen for the PRODUCT segment, the segment which contains the detail of the products sold for the data displayed.

Using Multiple Record Screens

Press PF5 at the Key Values Screen for the PRODUCT segment to display all the instances in the PRODUCT segment.

This screen is called the Multi-Record Data Screen for PRODUCT. Segment instances are retrieved in the order that they are sorted by their key field values. If the list of instances in this segment is too wide to be displayed on your screen, press PF8 to scroll to the fields on the right, and PF7 to scroll to the fields on the left. To display more segment instances, press PF5.

You can update the values of the non-key fields by moving to specific fields and entering the new values. Use the Tab and Arrow keys to move to any field on your screen. When you press Enter, the values on the screen are written back to the database.

You can delete or show instances using the D/S column. To show a single instance from the Multi-Record Data Screen, type the following value in the D/S column next to the instance's key value and press Enter:

```
s
```

The instance is displayed on a screen by itself. Then, press PF6 to return to the Multi-Record Data Screen.

To delete an instance, type the following value in the D/S column next to the instance's key value and press Enter:

```
d
```

The instance is displayed on a screen by itself and you are prompted to verify that you want this instance deleted. For this example, you can either choose to delete the instance or cancel the deletion. To delete the instance, press Enter. To cancel the deletion, press PF6. You are returned to the Multi-Record Data Screen.

Note: You cannot add new instances in Multi-Record Data Screens.

Now, to exit ModifyTalk and return to the FOCUS command prompt, press PF3. You can press PF3 to end the execution of your MODIFY procedure and return to the FOCUS command prompt at any point you wish.

The function keys available when using a FOCEXEC generated by ModifyTalk are listed in **Appendix A, *Overview of the Master File Description***.

Generating a Customized Maintenance Procedure

By generating customized procedures, you can tailor MODIFY FOCEXECs to suit the needs of your users. For example, if one user handles only data input while another performs a variety of maintenance actions, you can easily generate customized procedures to suit the needs of each one.

Defining functions for your MODIFY FOCEXEC is a simple task. ModifyTalk can enhance your whole application development cycle. With ModifyTalk, you can quickly prototype applications, try them out (or let the users try them out for you), then generate refined versions. The generated FOCEXEC can be used as is or be edited to add custom screens, validation tests, and other features that best adapt it to your needs.

In this section, we provide an overview of the steps and the screens involved in this process and show how they are used when creating your FOCEXEC. **In Tutorial: Two Sample FOCEXECs on page 5-36, we illustrate how the requirements of your application can be translated into the processing logic of your solution.**

Developing the Logic of the FOCEXEC

Before you describe a customized MODIFY FOCEXEC, you must plan the processing logic of the procedure. The most important step in planning your FOCEXEC is to decide what modifications you want to make to your data file. This enables you to specify the following to ModifyTalk:

- The file segments you want to activate, that is, update/add/view fields of the segment in the procedure.
- The actions you want to take for each segment when an instance is retrieved.
- The segment you want to process after a MATCH or NOMATCH action has been taken.

The answers to these questions outline the processing logic of your procedure and your ModifyTalk session. ModifyTalk prompts you for information in the same order.

Describing the Customized FOCEXEC

After you plan the processing logic, you then describe your customized MODIFY FOCEXEC by selecting options from a series of FOCEXEC Processing windows. The series of windows include the same five which describe a default procedure, plus three windows which describe matching conditions and control processing flow.

To begin describing a customized FOCEXEC, invoke ModifyTalk first and select a file from the list of databases. Then, you select:

1. The *Customized maintenance procedure* option in the Process Selection window.
2. Which segments to activate in the Segment Activity window. You may activate all of the segments or select individual segments from the list.
3. After you select the active segments, you can begin specifying the processing logic. ModifyTalk processes each segment in the data file by its logical order in the structure. You determine the MATCH, NOMATCH, and Control Flow selections for each

segment.

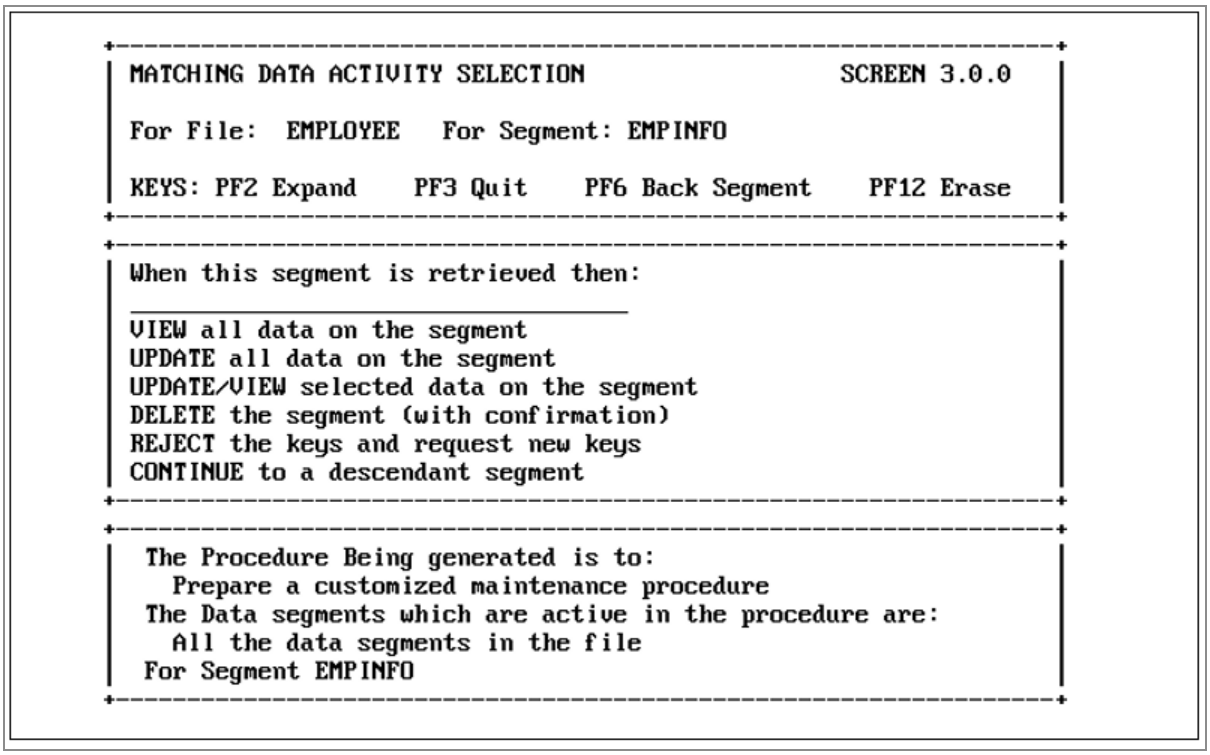
The FOCEXEC Processing windows for MATCH, NOMATCH, and Control Flow selections are explained in the following sections.

Note:

- Step-by-step instructions for accessing all of the windows used to create a customized FOCEXEC are provided in the tutorials in ***Tutorial: Two Sample FOCEXECs***.
- For information about the Processing windows used to generate a default procedure, see ***Tutorial: Generating a Default Procedure***.

Matching Data Activity Selection Window

You select MATCH actions from the Matching Data Activity Selection window. These are the actions to take when the MODIFY FOCEXEC retrieves a segment instance, as shown in the following image:

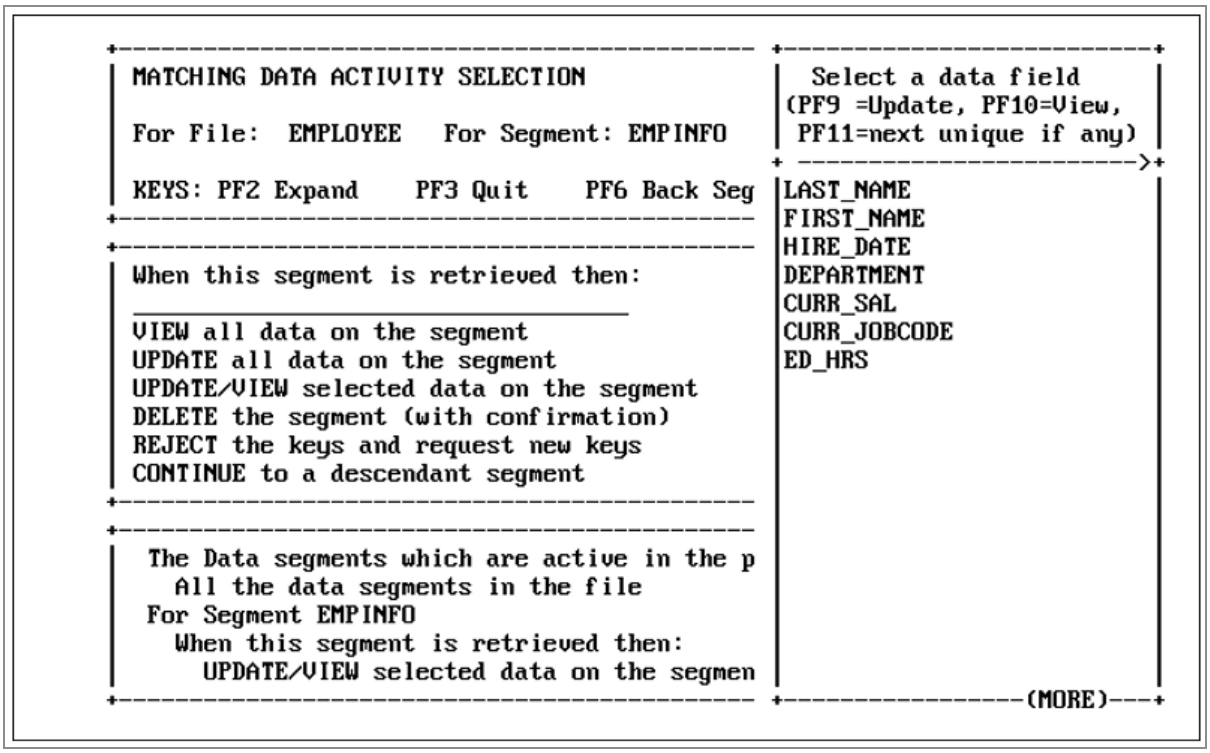


The options on this window enable you to select the following actions for your MODIFY procedure:

Option	Action
`VIEW`	FOCEXEC displays the values of all fields in the retrieved segment on the screen. This includes fields in unique segments if they are active. However, no changes can be made to the data.
`UPDATE`	FOCEXEC enables you to change and display the values of all fields in the retrieved segment, except the key field. This includes fields in active unique segments.
`UPDATE/VIEW`	FOCEXEC shows both display fields and update fields on the screen. You can select the fields that the FOCEXEC displays and specify the fields that can be modified.
`DELETE`	FOCEXEC deletes the matching segment instance from the file. All data in descendant segments of this instance are also removed. Because a deleted instance cannot be recovered, the FOCEXEC asks for verification before completing the action.
`REJECT`	FOCEXEC rejects the key value. The key screen is presented again to request another key value.
`CONTINUE`	FOCEXEC continues directly to a descendant segment. You specify a descendant segment in the Control Flow Windows that follow.

Note that unique segments are not automatically made active along with their parents. You must make them active when you are in the Segment Activity window, either by selecting all segments, or by specifically choosing the unique segment from a list.

When you choose the UPDATE/VIEW option for a segment, ModifyTalk automatically displays a window listing all its fields. You can then select the fields you would like to display and modify, as shown in the following image. The fields that are not selected are not displayed by the MODIFY FOCEXEC:



To select a field for display only, move the cursor to it and press PF10. The fieldname automatically appears on the right side of the bottom window, and you remain in the data field selection window for other selections.

To select a field for display with the option to update, move the cursor to it and press PF9. The fieldname automatically appears on the left side of the bottom window and you remain in the data field selection window for other selections.

To select from a unique segment, press PF11. The fieldnames in the parent segment are replaced by the fieldnames in the unique segment.

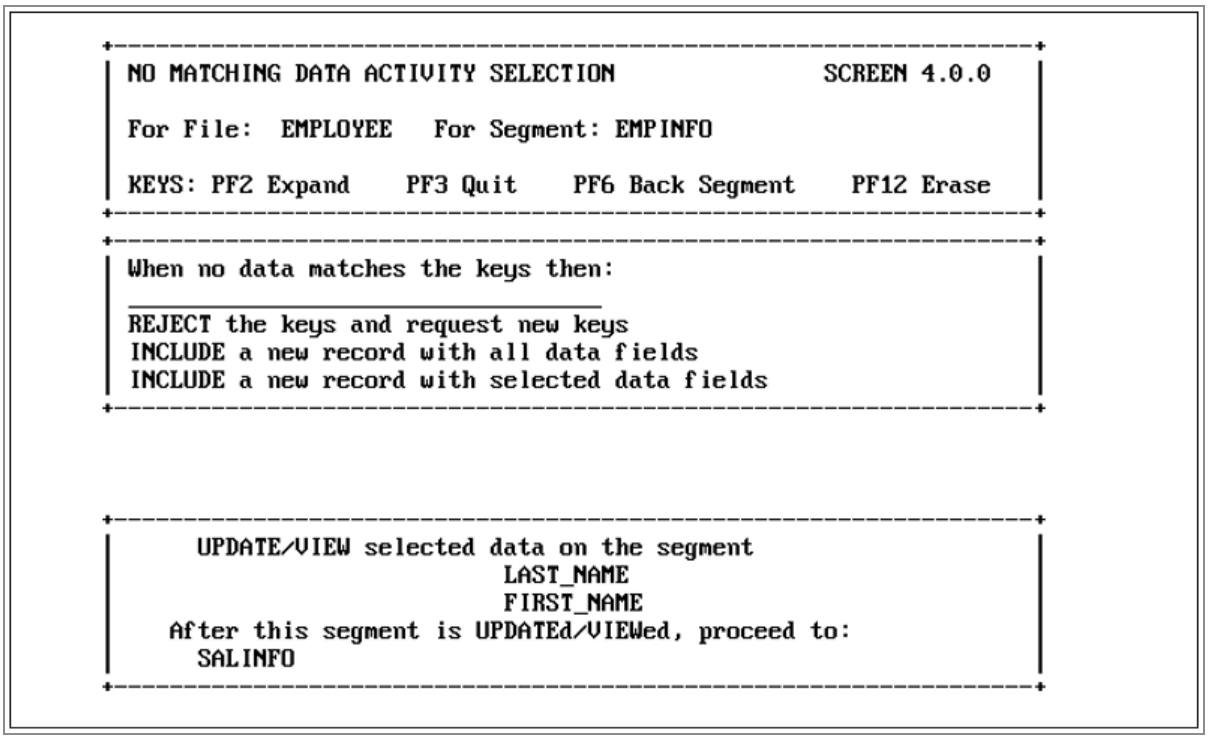
You can choose fieldnames in the descendant segments for display and update actions the way you would choose fieldnames in the parent segment.

Matching Data Activity Selection windows are usually followed by a Control Flow window. The kind of Control Flow window that follows depends on your choice of a MATCH action and your current location in the file.

No Matching Data Activity Selection Window

You select NOMATCH actions from the No Matching Data Activity Selection window. NOMATCH actions specify the actions that a MODIFY FOCEXEC takes when it is unable to

locate a segment instance, as shown in the following image:



The options in this window enable you to select the following actions for your MODIFY procedure:

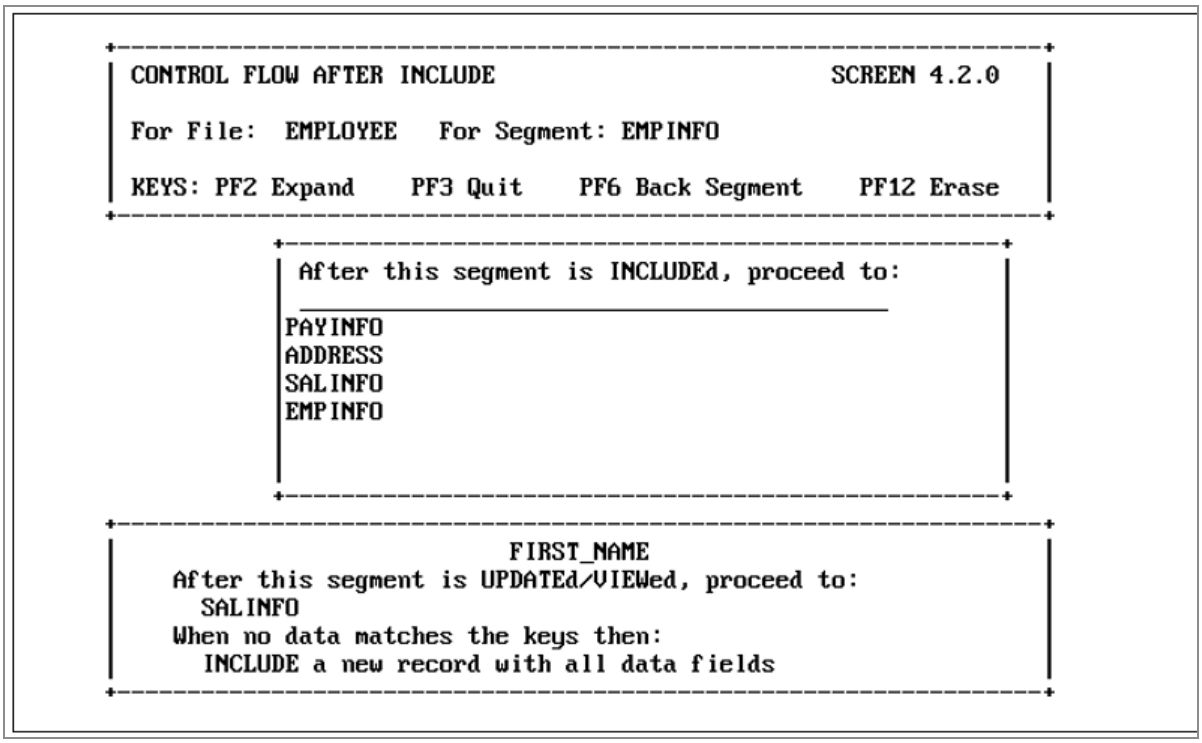
Option	Action
`REJECT`	FOCEXEC displays a message indicating that no matching record was found and prompts you for another key value.
`INCLUDE . . . all data fields`	FOCEXEC inputs a new instance for all fields in the data file. During execution, the names of all displayed fields in the segment are followed by a space where new values can be entered.
`INCLUDE . . . selected data fields`	FOCEXEC enters a new instance for selected fields in the data file. During execution, the names of all selected fields are displayed with a space where new values can be entered. You choose the fields to be entered after selecting this option. The procedure for choosing the fields is the same as that used for the UPDATE/VIEW option in the Matching Data Activity Selection window.

The No Matching Data Activity Selection window is usually followed by a Control Flow window. The specific window that follows depends on your choice of a NOMATCH action and on where you are in the file. Control Flow windows are described in Generating a Customized Maintenance Procedure.

Control Flow Windows

In these windows, you tell the FOCEXEC which segment to process next with options on the Control Flow window. This is equivalent to indicating what key field screen you want to see after each MODIFY action is complete.

The specific Control Flow window you see is titled *CONTROL FLOW AFTER action*, where *action* is the name of the MATCH or NOMATCH action you selected. For example, if you choose the option to include an instance, the Control Flow window that follows is called CONTROL FLOW AFTER INCLUDE, as shown in the following image:



Control Flow windows immediately follow the Matching and No Matching Data Activity Selection windows whenever your selected MATCH or NOMATCH actions enable you to continue to another segment.

The specific windows that follow each Matching and No Matching Data Activity Selection window depend on whether you are in the root segment, an intermediate segment, or the bottom segment. The tables that follow show the windows associated with each MATCH and NOMATCH option in the three segments.

From the root segment:

ACTIVITY SELECTION SCREEN WAS:	OPTION SELECTED WAS:	NEXT SCREEN IS:	DEFAULT CHOICE IS:
MATCH	View Update Update/View Delete Reject Continue	Control Flow Control Flow Control Flow NOMATCH NOMATCH Control Flow	current segment current segment current segment next active segment
NOMATCH	Reject Include	MATCH (for next segment) Control Flow	next active segment

Note:

- Selecting VIEW, UPDATE, or UPDATE/VIEW leads to the Control Flow window. The default Control Flow choice is the root segment. Other available segments are listed after it.
- Selecting DELETE or REJECT leads directly to the No Matching Data Activity Selection window. You are not offered a Control Flow choice because, when using the generated FOCEXEC, deleting or rejecting the matching instance leaves no active instances to continue to descendant segments.
- Selecting CONTINUE leads to the Control Flow window. The default Control Flow choice is the next lower logical segment. The other available segments are listed after it. If there is only one descendant, the Control Flow window is skipped.

- Selecting REJECT from the No Matching Data Activity Selection window leads directly to the Matching Data Activity Selection window for the next segment. You are not offered a Control Flow choice because, when using the generated FOCEXEC, rejecting the key field value does not create a new segment instance from which to continue to descendant segments in the file.
- Both INCLUDE options lead to the Control Flow window. The default Control Flow choice is the next lower logical segment. The other available segments are listed after it. If there is only one descendant, the Control Flow window is skipped.

From the intermediate segment:

ACTIVITY SELECTION SCREEN WAS:	OPTION SELECTED WAS:	NEXT SCREEN IS:	DEFAULT CHOICE IS:
MATCH	View Update Update/View Delete Reject Continue	Control Flow Control Flow Control Flow Control Flow Control Flow Control Flow	current segment current segment current segment current segment current segment next active segment
NOMATCH	Reject Include	Control Flow Control Flow	current segment next active segment

Note:

- Selecting VIEW, UPDATE, or UPDATE/VIEW leads to the Control Flow window. The default Control Flow choice is the current segment. Other available segments are listed after it.
- Selecting DELETE or REJECT leads to the Control Flow window. Because deleting or rejecting the matching instance leaves no active instance, you do not have the option

of continuing to descendant segments. The default Control Flow choice is to remain in the current segment. The other available segments are listed after it.

- Selecting CONTINUE leads to the Control Flow window. The default Control Flow choice is the next lower logical segment. The other available segments are listed after it. If there is only one descendant, the Control Flow window is skipped.
- Selecting REJECT from the No Matching Data Activity Selection window leads to the Control Flow window. Because rejecting the key field value does not create a new segment instance, you do not have the option of continuing to descendant segments. The default Control Flow choice is to remain in the current segment. The other available segments are listed after it.
- Both INCLUDE options lead to the Control Flow window. The default Control Flow choice assumes you want to add a new, complete record by continuing to the next lower logical segment. The other available segments are listed after it. If there is only one descendant, the Control Flow window is skipped.

From the bottom segment:

ACTIVITY SELECTION SCREEN WAS:	OPTION SELECTED WAS:	NEXT SCREEN IS:	DEFAULT CHOICE IS:
MATCH	View Update Update/View Delete Reject Continue	Control Flow Control Flow Control Flow Control Flow Control Flow Error Message	current segment current segment current segment current segment current segment current segment
NOMATCH	Reject Include	Control Flow Control Flow	current segment current segment

Note:

- From segments at the bottom of the file structure all choices, except CONTINUE, lead to the Control Flow window. Since there are no lower-level segments, the default Control Flow choice is the current segment. The other available segments are listed after it.
- Because there are no lower-level segments, you cannot choose CONTINUE. If you try to select CONTINUE, ModifyTalk displays an error message and returns you to the Matching Data Activity Selection window to select another option.

Finishing the Customized FOCEXEC

Now that you used the Processing Logic windows for each active file segment, complete your description of the customized FOCEXEC with the remaining Processing windows:

- In the Multiple Record Selection window, you choose whether or not multiple record screens should be available.
- In the End of Generation Selection window, you choose whether to save the generated procedure, or execute it without saving.
- In the Saved Procedure Selection window, you indicate how to process the procedure after saving.

After you specify whether or not to save the procedure, and determine what to do with the saved version, ModifyTalk generates the procedure you described.

As the procedure is being created, ModifyTalk displays a series of messages describing each step of this process, as shown in the following image:

```
Generating screens
Generating data handling logic
Generating key selection logic
Generating match logic
Generating the procedure
Procedure saved as EMP1
```

If you chose EXECUTE or COMPILE AND RUN from the Saved Procedure Selection window, ModifyTalk immediately runs your new procedure. When you finish using the MODIFY procedure, you are returned to the FOCUS command prompt.

Revising the Customized FOCEXEC

You can further customize the MODIFY procedure by adding computations and data validations, as well as customized messages or screens using TED, the FOCUS editor. However, unless you are an experienced FOCUS user, you should not change any of the processing logic; this could cause unpredictable results during execution. If you want to change the procedure's logic, you should use ModifyTalk to generate another FOCEXEC.

If you look at the MODIFY procedure with an editor, notice that the procedure begins with text describing the logic of the MODIFY routine. This is the text that was displayed in the bottom window of all the screens in your ModifyTalk session, and it serves to document the procedure.

ModifyTalk FOCEXEC Cases and Variables

FOCEXECs generated by ModifyTalk make extensive use of Case Logic. Cases used in a FOCEXEC are listed in the following chart, along with descriptions of their functions. Except for the two special cases at the end of the list, END.CHAIN and TYPE.MSG, all the cases follow the naming convention

```
command.Snn
```

where:

command

Is a name that indicates the function performed.

nn

Is a two digit number corresponding to the active file segment.

For example, a case that deletes segment instances for the third active file segment would be named:

```
DELETE.S03
```

Case Names

Case Name	Function
`top case`	Declares the local variables.
`KEYCRT.Snn`	Accepts either a key value or a PF key. Checks that the choice is valid and takes appropriate action.

Case Name	Function
`MATCH.Snn`	Establishes a position in the file and determines which action to take for the current segment. The variable SWT_CRNT indicates the current segment number. In the case of a NOMATCH, a message is displayed. The message displayed depends on whether or not the requested action was an INCLUDE.
`GETNEXT.Snn`	Gets the next record in the chain and displays it on the key field or existing data screen.
`TOPNEXT.Snn`	Establishes a position in the file for multiple record processing. If no initial position was established, repositions to the beginning of the chain.
`NEXT.Snn`	Initializes the stack and gets the first record of 12. At the end of chain, checks variable SWT_CHAIN to see whether or not the chain is empty. If chain is empty, SWT_CHAIN will be 99.
`RETRIEV.Snn`	Gets remaining records for multiple record processing. Loops until a count of 12 or the end of the chain is reached.
`REMATCH.Snn`	Establishes a starting position for multiple record processing if a key value was provided. Re-establishes the position after a group of 12 is displayed.
`ENDRETR.Snn`	Processes the end of multiple record retrieval. Checks SWT_CHAIN to see whether or not the chain is empty. If chain is empty, SWT_CHAIN will be 99. If the

Case Name	Function
	chain is empty, breaks out of multiple record processing.
`DISPLAY.Snn`	Displays 12 instances on the multi-record data screen and processes them. For entries to the D/S Control Column, performs the appropriate delete or update case.
`UPDATE.Snn`	Displays and updates a single instance, either directly from the key field screen or performed from the multi-record data screen.
`UPDNEXT.Snn`	Gets the next instance for the update screen. If no next instance, goes back to key field or multi-record data screen.
`DELETE.Snn`	Deletes segment instance from the multi-record data screen. Prompts for confirmation before completing the deletion.
`INCLUDE.Snn`	Accepts data for a new instance and adds it to the file. Allows instance to be skipped in case of errors.
`END.CHAIN`	Determines whether or not instances exist in the chain.
`TYPE.MSG`	Displays general error messages.

Variables

FOCEXECs generated by ModifyTalk use a standard set of variables. The variables that can be used in a FOCEXEC are listed in the following table, along with descriptions of their functions.

Variable Name	Function
`SWT_CURNT`	Indicates the current segment number.
`SWT_COUNT`	Indicates the number of records held for multiple record procedures.
`SWT_CHAIN`	Indicates whether chain is empty.
`SWT_MSG`	Displays message when end of chain is reached.
`SWT_VAL`	Checks PF key validity.
`SWT_PRFRM`	Determines whether current case is called by PERFORM or GOTO in UPDATE, UPDNEXT, or DELETE.
`SWT_CTL`	Indicates whether D or S option is active in multi-record data screen.

Function Keys for FOCEXECs Created With ModifyTalk

The following topic describes the function keys that are available in a ModifyTalk procedure.

Function Keys

Function Key	Purpose
PF3	Stops the FOCEXEC. This key is always available during the procedure.
PF4	Shows the next instance for the segment.
PF5	Shows the multiple record screen for the segment.
PF6	On multiple record screens, it returns to the key value screen. • On new data input screens, it rejects the supplied key value. • On delete instance screens, it cancels the delete choice. • On existing data display screens, it skips the unique segment.
PF7	On multiple record screens, it shows more fields to the left.
PF8	On multiple record screens, it shows more fields to the right.
PF9	Returns to the key field screen for the preceding segment.
PF10	Adds the key field screen for the next segment.
PF11	For multiple path files, it selects the next segment to the right.

ModifyTalk-Generated FOCEXEC

ModifyTalk is extensively self-documenting. When your FOCEXEC is generated, the description of the procedure is copied from ModifyTalk's bottom window to the top of the FOCEXEC. This gives you a thorough description of the whole procedure. In addition, each case in the FOCEXEC is preceded by comments that describe its function.

Sample ModifyTalk-Generated FOCEXEC

The following code shows an example of the default code that is generated by ModifyTalk. The actual MODIFY commands have been removed, leaving only the comment lines and case names. Reviewing this, you can see how a typical ModifyTalk-generated FOCEXEC is organized and commented.

```
-*************************************************************************
-* M O D I F Y T A L K G E N E R A T E D P R O C E D U R E
-* For File SALES ON 06/17/92 AT 15.31.34 BY DOCRDM
-*The Procedure Being generated is to:
-*Prepare a procedure with all actions defaulted
-* The Data segments which are active in the procedure are:
-*All the data segments in the file
-* Allow multiple record processing:
-*For all segments in the file
-* At the end of the generation:
-*Save the procedure generated
-* After saving the procedure:
-*Execute the saved procedure
-*************************************************************************
MODIFY FILE SALES
COMPUTE
SWT_CURNT/I2=; SWT_MSG/A70=; SWT_VAL/I2=; SWT_CTL/A1=;
SWT_COUNT/I5=; SWT_CHAIN/I2=; SWT_PRFRM/I1=; PFKEY/A4=;
HOLDINDEX/I5=;
HOLDINDEX/I5=;
STORE_CODE/A3=;
DATE/A4MD=;
PROD_CODE/A3=;
GOTO KEYCRT.S01
-*
-*************************************************************************
-* The following section handles accepting key values from the user, *
-* allowing minimal navigation within the file, and multi-record *
-* signaling. Keys are matched later, and the action proceeds according*
-* to the match/nomatch pattern established. *
-*************************************************************************
-* Accept a key from the user, or one of a set of control pfkeys. Check
-* the validity of the choice made, and act accordingly.
-*
CASE KEYCRT.S01
```

```
ENDCASE
-*
-* Accept a key from the user, or one of a set of control pfkeys. Check
-* the validity of the choice made, and act accordingly.
-*
CASE KEYCRT.S02
ENDCASE
-*
-* Accept a key from the user, or one of a set of control pfkeys. Check
-* the validity of the choice made, and act accordingly.
-*
```

```
CASE KEYCRT.S03
ENDCASE
-*
-* Establish a position within the file. The match is done on every
-* level, thus sometimes going through more than one match case.
-* SWT_CURNT indicates the segment number which is current, at which
-* level action should be taken. In case of a nomatch, a message is
-* displayed depending on whether or not there was an include requested
-* at that level.
-*
CASE MATCH.S01
ENDCASE
-*
-* Establish a position within the file. The match is done on every
-* level, thus sometimes going through more than one match case.
-* SWT_CURNT indicates the segment number which is current, at which
-* level action should be taken. In case of a nomatch, a message is
-* displayed depending on whether or not there was an include requested
-* at that level.
-*
CASE MATCH.S01
ENDCASE
-*
-* Establish a position within the file. The match is done on every
-* level, thus sometimes going through more than one match case.
-* SWT_CURNT indicates the segment number which is current, at which
-* level action should be taken. In case of a nomatch, a message is
-* displayed depending on whether or not there was an include requested
-* at that level.
-*
CASE MATCH.S02
ENDCASE
-*
-* Get the next record in the chain to display on the key screen.
```

```
-*
CASE GETNEXT.S02
ENDCASE
-*
-*
CASE MATCH.S03
-*
-* Get the next record in the chain to display on the key screen.
-*
CASE GETNEXT.S03
ENDCASE
-*
```

```
-*************************************************************************
-* The following section handles the action at the segment data level.
* ENDCASE
-*
-* If no initial position was given in the key screen, reposition.
-*
CASE TOPNEXT.S01
ENDCASE
-*
-* Initialize the stack, and get the first record of 12. At end of chain
-* SWT_CHAIN indicates whether the chain is empty (99 is the value).
-*
CASE NEXT.S01
ENDCASE
-*
-* Loop and get remaining records until 12 or end of chain are reached.
-*
CASE RETRIEV.S01
ENDCASE
-*
-* Get direct position if a key was given in the key screen, or to
-* re-establish the position after a group of 12 was displayed.
-*
CASE REMATCH.S01
-* (SWT_CHAIN is 99), break out.
-*
CASE ENDRETR.S01
ENDCASE
-*
-* Display 12 records and process them. In case of 'D/S', perform
-* the delete or update case accordingly.
-*
CASE DISPLAY.S01
```

```
ENDCASE
-* or performed from the multi-record screen).
-*
CASE UPDATE.S01
ENDCASE
-*
-* Display and update a single record (directly from the key screen,
-* or perform PDATE * SEG 1
ENDCASE
-*
-* Get the next record to display in update mode. If none left,
-* get back to multi-record screen or to the key screen.
-*
CASE UPDNEXT.S01
ENDCASE
-*
-* Ask for confirmation, and delete a record (directly from the key
-* screen, or performed from the multi-record screen).
-*
```

```
CASE DELETE.S01
ENDCASE
-*
-* Accept data for a new record, and add it to the file. Allow skipping
-* of the record for cases of errors.
-*
CASE INCLUDE.S01
ENDCASE
-*
-*
-* Establish position in the file for multi record processing
-* If no initial position was given in the key screen, reposition.
-*
CASE TOPNEXT.S02
ENDCASE
-*
-*
-*
-* Loop and get remaining records until 12 or end of chain are reached.
-*
CASE RETRIEV.S02
ENDCASE
-*
-* Get direct position if a key was given in the key screen, or to
-* re-establish the position after a group of 12 was displayed.
-*
```

```
CASE REMATCH.S02
ENDCASE
-*
-*
CASE ENDRETR.S02
ENDCASE
-*
-* Display 12 records and process them. In case of 'D/S', perform
-* the delete or update case accordingly.
-*
CASE DISPLAY.S02
-*
-* Processing for end of record retrieval. If none in the chain
-* (SWT_CHAIN is 99), break out.
ENDCASE
-*
-* Display and update a single record (directly from the key screen,
-* or performed from the multi-record screen).
-*
CASE UPDATE.S02
ENDCASE
-*
-* Get the next record to display in update mode. If none left,
-* get back to multi-record screen or to the key screen.
-*
```

```
CASE UPDNEXT.S02
ENDCASE
-*
-* Get the next record to display in update mode. If none left,
-* get back to multi-record screen or to the key screen.
-*
-*
-* Accept data for a new record, and add it to the file. Allow skipping
-* of the record for cases of errors.
-*
CASE INCLUDE.S02
ENDCASE
-*
-* Accept data for a new record, and add it to the file. Allow skipping
-* of the record for cases of errors.
-*
-* Establish position in the file for multi record processing
-* If no initial position was given in the key screen, reposition.
-*
CASE TOPNEXT.S03
```

```
ENDCASE
-*
-* Initialize the stack, and get the first record of 12. At end of chain
-* SWT_CHAIN indicates whether the chain is empty (99 is the value).
-*
-* Loop and get remaining records until 12 or end of chain are reached.
-* SWT_CHAIN indicates whether the chain is empty (99 is the value).
-*
CASE NEXT.S03
ENDCASE
-*
-* Get direct position if a key was given in the key screen, or to
-* re-establish the position after a group of 12 was displayed.
-*
CASE REMATCH.S03
-* Loop and get remaining records until 12 or end of chain are reached
ENDCASE
-*
-* Processing for end of record retrieval. If none in the chain
-* (SWT_CHAIN is 99), break out.
-*
CASE ENDRETR.S03
ENDCASE
-*
-*
CASE DISPLAY.S03
-*
-* Display 12 records and process them. In case of 'D/S', perform
ENDCASE
-*
-* Display and update a single record (directly from the key screen,
-* or performed from the multi-record screen).
-*
```

```
CASE UPDATE.S03
ENDCASE
-*
-* Get the next record to display in update mode. If none left,
-* get back to multi-record screen or to the key screen.
-*
CASE UPDNEXT.S03
ENDCASE
-*
-* Ask for confirmation, and delete a record (directly from the key
-* screen, or performed from the multi-record screen).
-*
```

```
CASE DELETE.S03
ENDCASE
-*
-* Accept data for a new record, and add it to the file. Allow skipping
-* of the record for cases of errors.
-*
CASE INCLUDE.S03
ENDCASE
-*
-* General error/message display.
-*
CASE TYPE.MSG
ENDCASE
-*
DATA VIA FI3270
END
```

Terminal Operator Environment

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

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

These topics use the EMPLOYEE data source in the examples. For more information about this data source, see the *ibi™ FOCUS Creating Reports* manual.

Invoking the Terminal Operator Environment

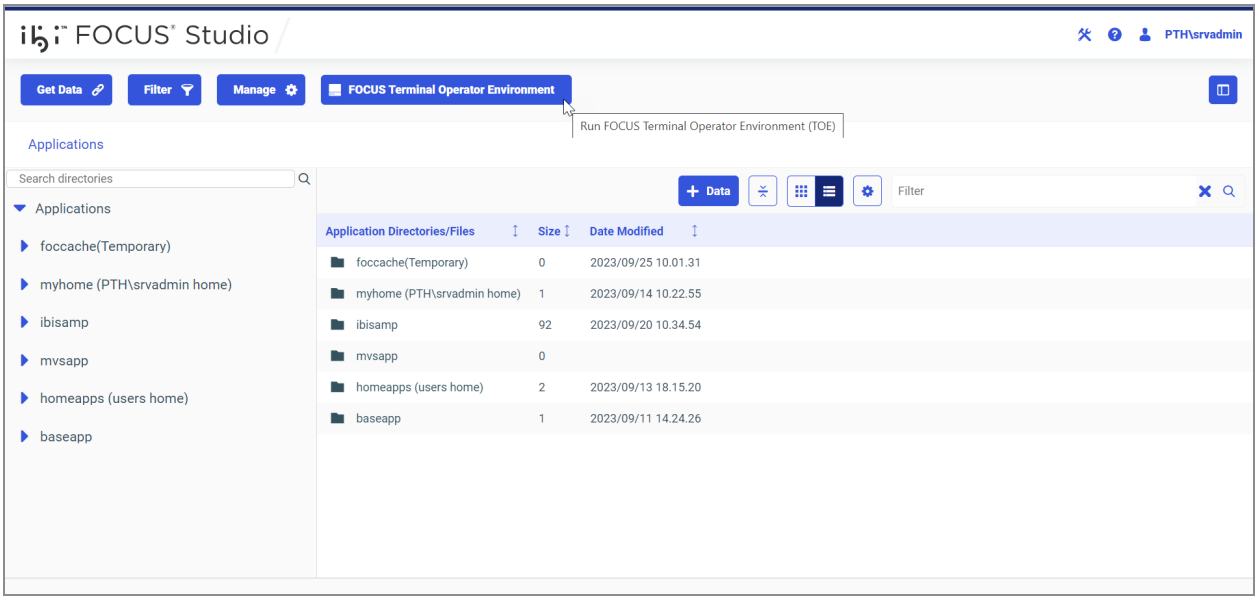
You can invoke the Terminal Operator Environment in either of the following ways:

- Using the FOCUS Studio console.
- Issuing the WINDOW ON command from the FOCUS command level.

The following image illustrates the Terminal Operator Environment.

The blank Command Window is available for commands and requests. The Output Window displays the input and the resulting output. The History Window has more than one screen of recorded commands and requests. Using predefined program function keys (PF keys), you can move around the screen to activate a window, enlarge a window to full screen size, or scroll window contents. You can also use WINDOW commands to control window behavior and to customize your screen.

To invoke the Terminal Operator Environment from the FOCUS Studio console, click **FOCUS Terminal Operator Environment**, as shown in the following image.



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

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

Enter the Terminal Operator Environment

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

where:

ON

Invokes the Terminal Operator Environment.

OFF

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

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

Activating a Window

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

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

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

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

Types of Windows

The Terminal Operator Environment consists of the following windows. Each window performs a function or accepts certain activities.

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

Command Window

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

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

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

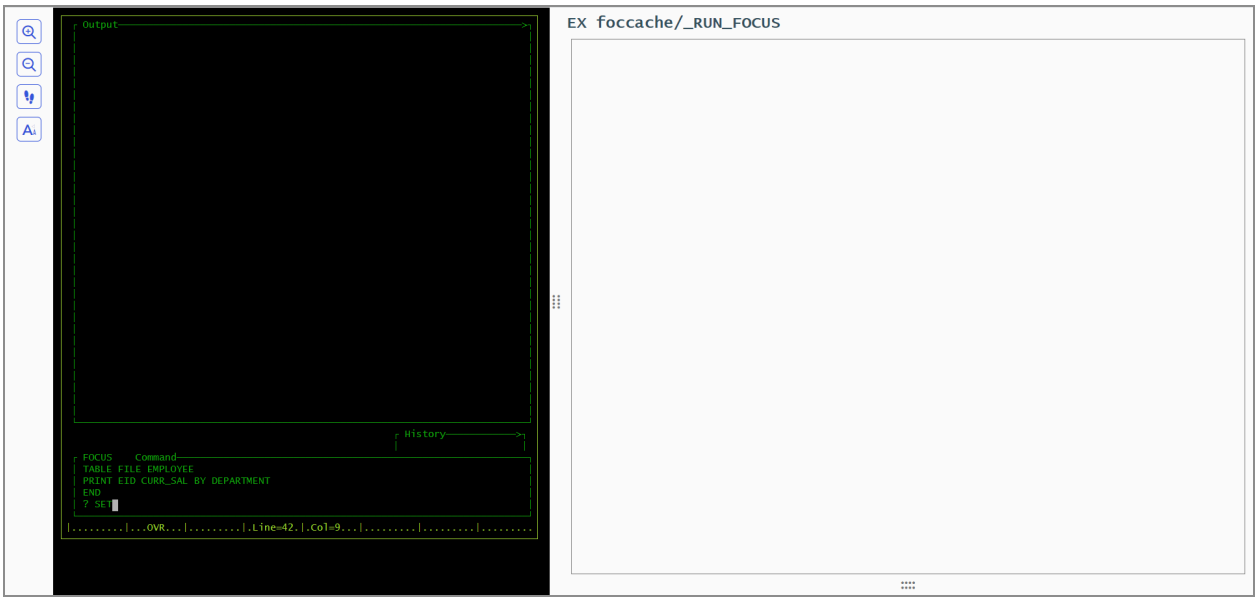
If your request is longer than four lines, press **PF2** to enlarge the window. Finish entering the rest of the request; do not press PF2 again. Press **Enter**. The complete request is copied to the Output Window and to the History Window and is submitted for execution.

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

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

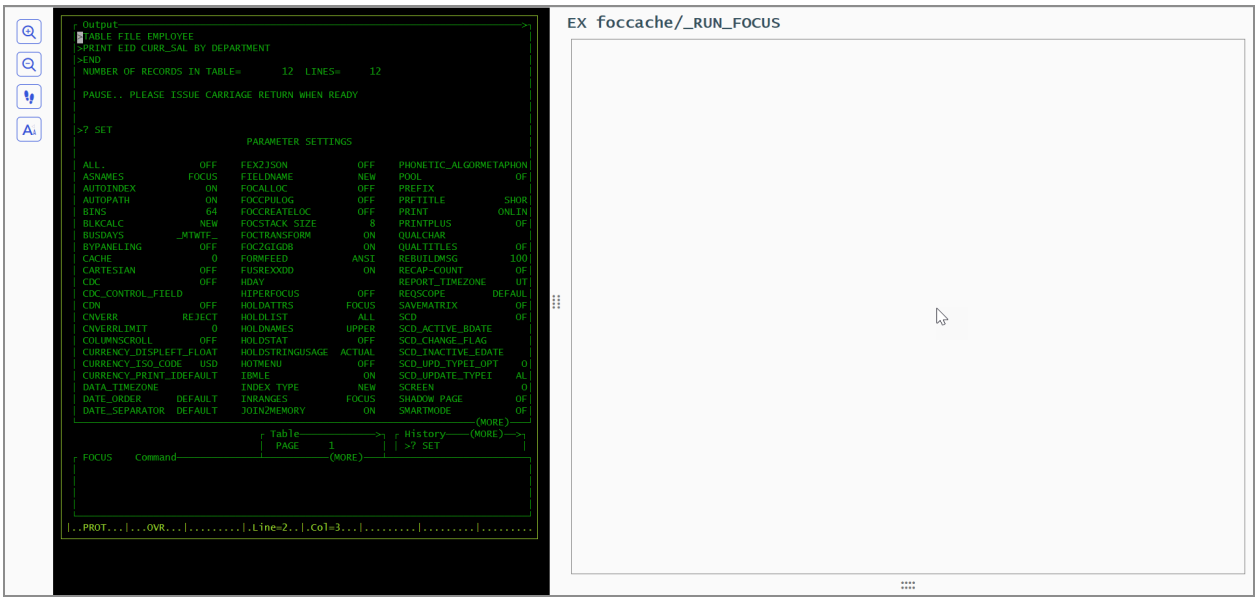
If you enter four FOCUS commands (each on a separate line) or a command and request (totaling four lines), the first item (command or request) is copied to the appropriate windows, submitted, and processed and followed by the next item. The Output and History Windows display each item in succession.

In the following image, the Command Window contains a combination report request and a query command.



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

The following image illustrates the Output Window after HotScreen displays the report.



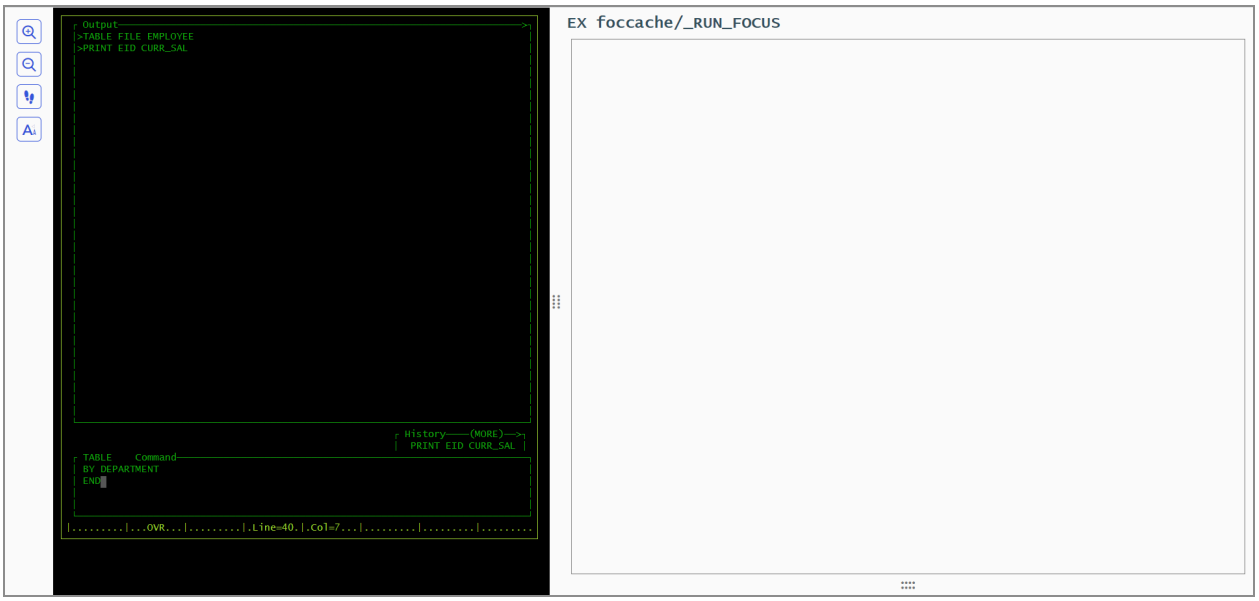
The Command Window also displays the current command mode in a title area. The title area is on the left side of the window's top border. It functions like the prompt does in the default FOCUS environment. In the previous screen sample, for example, the title area displays

```
FOCUS  Command
```

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

```
TABLE  Command
```

In the following image, part of the TABLE request has been submitted. The title area indicates a current command mode of TABLE.



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

Output Window

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

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

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

History Window

The History Window provides a history of your FOCUS session. It records commands entered from the Command Window. You can review up to 40 previously typed command

lines. For example, you could refer to it to see how you specified an earlier TABLE request.

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

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

Help Window: Revising PF Key Settings

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

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

The default key settings are:

Key	Value
PF1, PF13	Help
PF2, PF14	Zoom
PF3, PF15	
PF4, PF16	Scroll Top
PF5, PF17	Scroll Bottom
PF6, PF18	Recall
PF7, PF19	Scroll Backward
PF8, PF20	Scroll Forward

Key	Value
PF9, PF21	Move Cursor
PF10, PF22	Scroll Left
PF11, PF23	Scroll Right
PF12, PF24	Next

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

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

Erasing the Window Contents

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

```
PF3 CLEAR OUTPUT
```

Assigning WINDOW Commands as Key Settings

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

```
PF3 FOCUS ? SET
```

Note:

- You may also specify the WINDOW SET command from the Command Window to change a key setting for the session.

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

Table Window

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

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

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

Error Window

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

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

At this point, you have two choices:

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

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

Displaying Fields and Field Formats

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

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

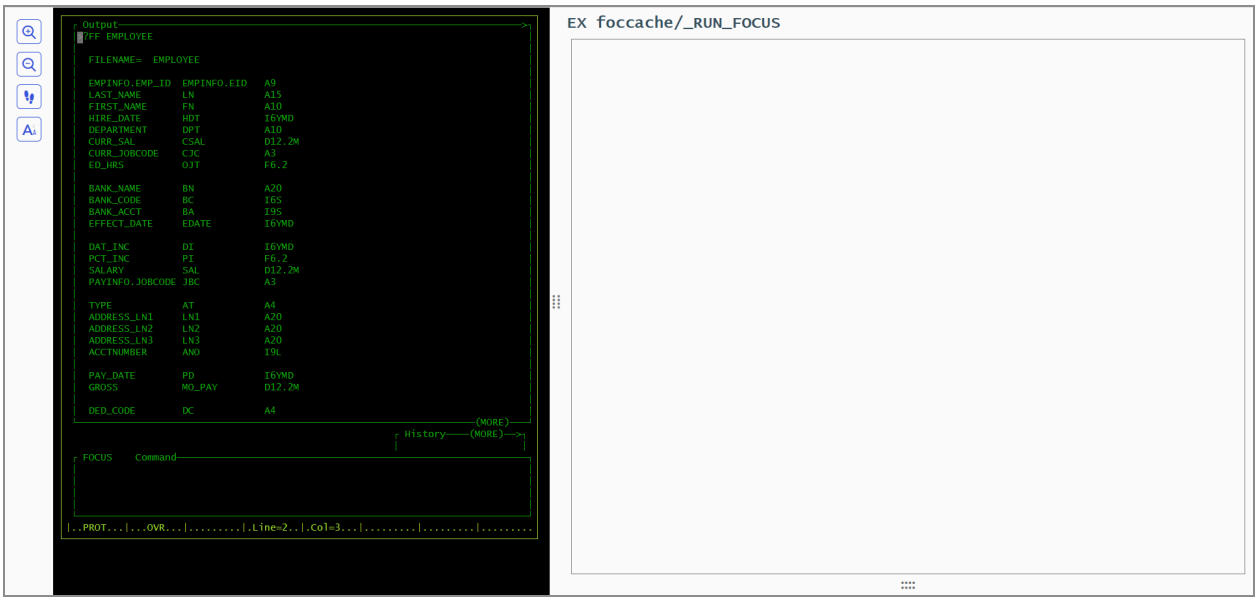
```
?FF
```

```
?F
```

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

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

In the following screen, the ?FF EMPLOYEE command was entered in the Command Window. When you press Enter, the Output Window displays the list of fields, as shown in the following image.



Window Commands

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

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

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

- The ACTIVE and NEXT commands activate a window.
- The CLEAR command clears the contents of a window.
- The SET CONTINUE, SET AUTOSCROLL, and SET IMMEDTYPE commands control the behavior of the Output Window.
- The CLOSE, OPEN, MOVE, SET ERRORS, SET, and SIZE commands customize your screen.
- The HELP command displays the Help Window.

- The ZOOM command enlarges a window.
- The RECALL command recalls issued commands.
- The ROUTE command routes the contents of a window to a data set.
- The SCROLL command controls the display of the window contents.

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

Commands for Activating a Window

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

Activate a Window

```
WINDOW ACTIVE windowname
```

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

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

```
WINDOW ACTIVE HISTORY
```

Activate the Next Window in the Screen Sequence

```
WINDOW NEXT
```

Pressing PF12 is equivalent to issuing the NEXT command.

Clearing a Window

The CLEAR command erases the contents of a window.

Erase the Contents of a Window

```
WINDOW CLEAR [windowname]
```

For example, to clear the Output Window, enter:

```
WINDOW CLEAR OUTPUT
```

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

Controlling the Output Window

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

Automatically Scroll the Contents of the Output Window

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

where:

ON

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

OFF

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

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

```
WINDOW SET AUTOSCROLL OFF
```

Control Data Transmission

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

where:

ON

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

OFF

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

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

```
WINDOW SET CONTINUE ON
```

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

Control Buffering of Line Mode Output

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

where:

ON

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

OFF

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

Customizing Your Screen

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

```
SET SBORDER=OFF
```

before entering the Terminal Operator Environment.

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

- The CLOSE command removes a window from the screen.
- The OPEN command displays a closed window in its normal screen location.
- The MOVE command moves a window to a new screen location.
- The SET ERRORS command controls the length of the error message that displays in the Error Window.
- The SET command is used to redefine key settings.
- The SIZE command changes the height or width of a window.

Remove a Window From the Screen

```
WINDOW CLOSE [windowname]
```

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

```
WINDOW CLOSE HISTORY
```

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

Display a Closed Window in its Normal Screen Location

```
WINDOW OPEN [windowname]
```

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

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

```
WINDOW OPEN HISTORY
```

Move a Window to a New Screen Location

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

where:

location

Is one of the following:

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

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

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

RIGHT same as LEFT, but to the right.

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

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

n

Is any positive number.

*****

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

For example,

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

```
WINDOW MOVE HISTORY ROW 10
```

To move the Table Window up 12 rows, enter:

```
WINDOW MOVE TABLE UP 12
```

Note:

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

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

```
MOVE [windowname] CURSOR
```

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

Control the Length of an Error Message in the Error Window

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

where:

SHORT

Displays the short form: the error number and description.

LONG

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

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

```
WINDOW SET ERRORS SHORT
```

Redefine Key Settings

```
WINDOW SET key command
```

where:

key

Is any PF key.

command

Is any WINDOW or FOCUS command.

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

```
WINDOW SET PF14 CLOSE
```

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

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

Note:

- You can edit the Help Window and immediately change the key settings (see Help Window: Revising PF Key Settings).
- Key settings change back to their default settings when you end the FOCUS session unless they are defined in the PROFILE FOCEXEC.

Change the Height or Width of a Window

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

where:

WIDTH

Changes the width of the window.

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

HEIGHT

Changes the height of the window.

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

n

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

*****

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

MORE

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

LESS

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

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

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

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

```
WINDOW SIZE HISTORY HEIGHT 5
```

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

Displaying the Help Window

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

Display the Help Window

```
WINDOW HELP
```

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

Enlarging a Window

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

Enlarge a Window

```
WINDOW ZOOM [windowname]
```

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

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

Recalling Commands

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

```
RECALL
```

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

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

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

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

Routing Window Contents

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

Route Window Contents

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

where:

ddname

Is any valid ddname.

OFF

Will stop routing data to the ddname.

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

```
WINDOW ROUTE HISTORY TO SESSION
```

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

Scrolling Window Contents

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

Scroll Window Contents

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

where:

direction

Is one of the following:

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

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

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

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

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

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

Note:

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

Left Icon Options

The following table lists the options that are available from the icons on the left of the full screen output panel.

Option	Description
	Zooms out the panel.
	Zooms in the panel.
	Displays low-level tracing. The default for tracing is off.
	Toggles between wide terminal (UTF) or narrow terminal display.

Designing Windows With Window Painter

The following topics describe how to create FOCUS menus and windows that work with FOCEXECs.

Introduction

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, and you can implement horizontal menus and multi-input windows as part of your FOCUS application. Horizontal menus can also have pull-down 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 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.

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 each to fit your application. You can design user-friendly menus and display convenient and eye-catching instructions onscreen.

FOCUS Window Painter itself guides you step by step, using windows like those you created.

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 written 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 z/OS.

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 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 command -WINDOW in the FOCEXEC. The -WINDOW command provides the name of the window file, and the name of the individual window that should be displayed first. When the -WINDOW command 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 command 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 command 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 Managing Flow of Control in an Application. For details of integrating a FOCEXEC with the Window facility using return and goto values, see Integrating Windows and the FOCEXEC.

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.

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 temporarily allocates the PDS. For a full description of allocation requirements, see the appropriate *Guide to Operations* topic 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.

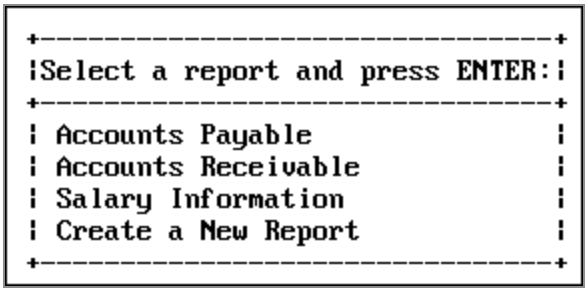
You can make an application more attractive by presenting menus in windows containing titles and other design elements, and can make an application easier to use by displaying function key definitions or other useful information.

Types of Windows You Can Create

Window Painter creates 10 different types of windows, each with its own special uses. These windows are described in the following topics.

Vertical Menus

This is a *vertical* menu:



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 Painter Screens.

Horizontal Menus

This is a *horizontal* menu:

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 pull-down 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.

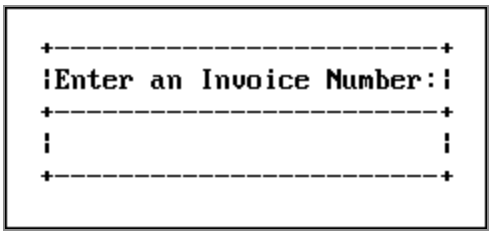
An application can display an associated pull-down menu for an item on a horizontal menu when the cursor is on that item. Choose the pull-down option from the Window Options menu as discussed in Window Files and Windows. An option to display descriptive text above or below the horizontal menu is also 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.

Text Input Windows

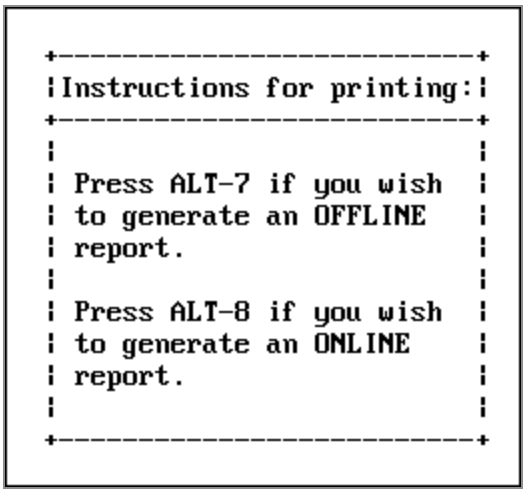
This is a *text input* window:



Amper variables can be used in a Windows application. A text input window prompts the user to supply information needed in a FOCEXEC. 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 Painter Screens.

Text Display Windows

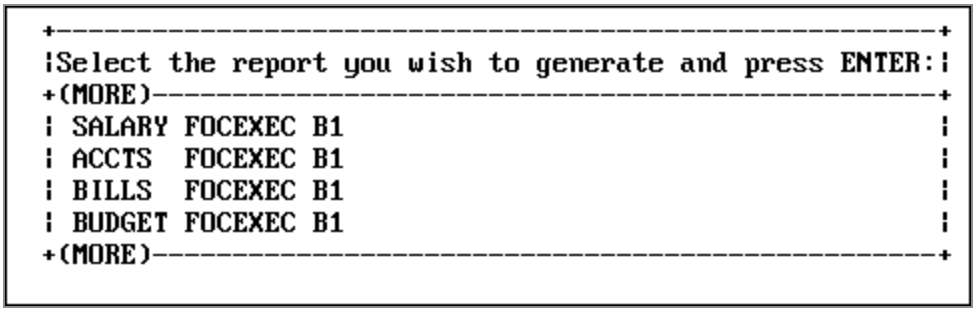
This is a *text display* window:



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.

File Names Windows

This is a *file names* window:

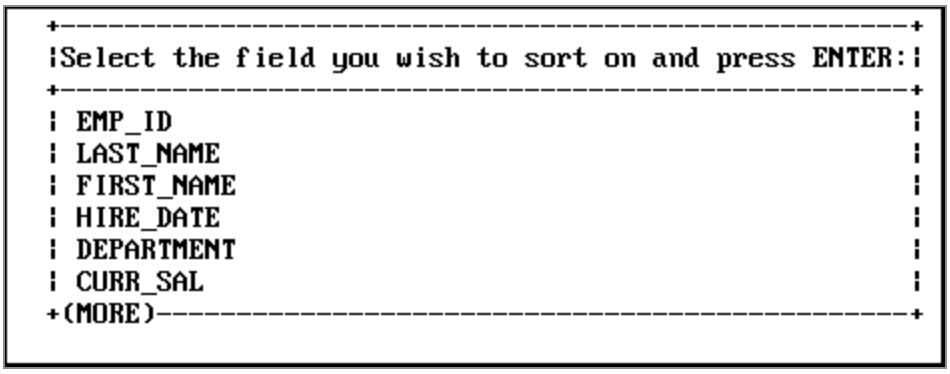


A File Names window presents a list of names of up to 1023 PDS members. The user can select one of these names by moving the cursor and pressing **Enter** when the cursor is on the line of the desired file name. You can specify selection criteria for the displayed file names when the window is created. 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. Narrow windows can display a greater number of names.

Field Names Windows

This is a *field names* window:



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.

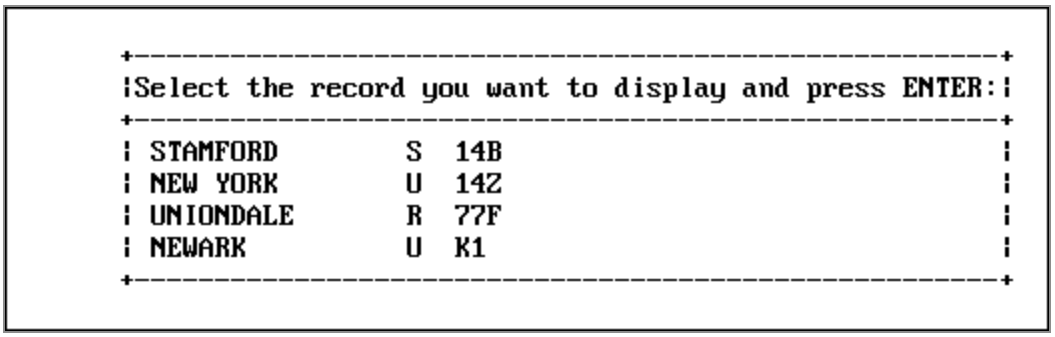
The field names are qualified when duplicates exist. You can use **PF10** and **PF11** to scroll left and right if a field name exceeds the maximum number of characters allowed on a line in a data field window.

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.

File Contents Windows

This is a *file contents* window:



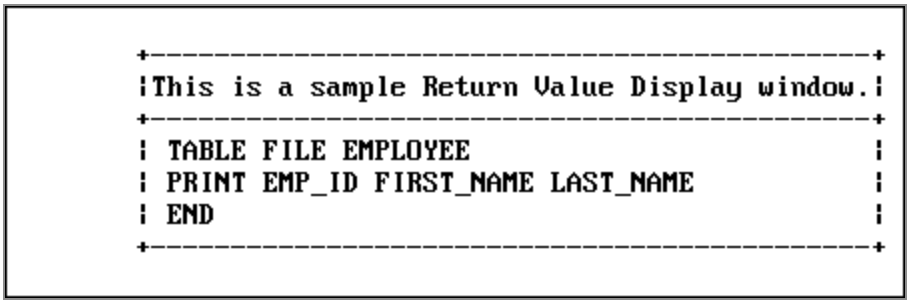
The file contents window displays the contents of a file. There is no limit on the size of a file contents window. The user can select a line of contents by moving the cursor to it and pressing **Enter**. Each line can be up to 77 characters long. A user can select more than one line if the window includes the Multi-Select option, which is described as part of the Window Options Menu in Window Painter Screens.

The contents of any member of a PDS (except as noted below) can be displayed. Sequential files can also be displayed in TSO. You are 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.

Return Value Display Windows

This is a *return value display* window:



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 Painter Screens.

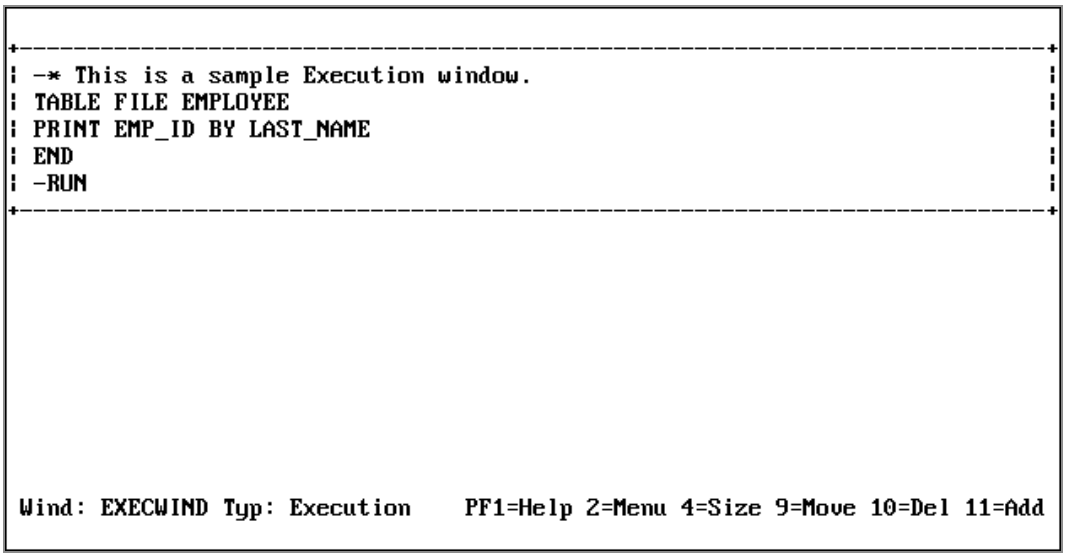
Only one return value display window may be displayed at a time on the screen. It collects a value from any active window (that is, a window from which a selection is being made or to which text is being entered, or an active text display window) if it is on that window's display list. A description of the Display lists option on the Window Options Menu can be found in Window Painter Screens.

You can clear the collected values from a return value display window by including it on the hide list of a window that is being used. A description of the Hide lists option on the Window Options Menu can be found in Window Painter Screens.

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.

Execution Windows

This is an *execution* window:



The execution window contains FOCUS commands such as Dialogue Manager commands, 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 that enables 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:

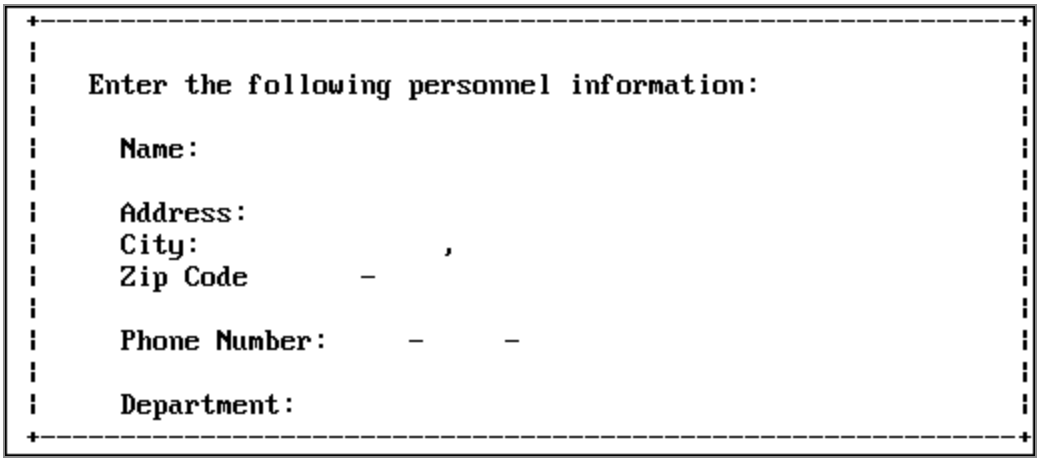
- When you GOTO an execution window, the contents of the window are executed. In all cases, execution begins at the top of the window.
- An execution window is not displayed when executed, although the commands it contains may generate a display.
- An execution window can use an amper variable as a goto 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 commands 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 follow the logic determined by the windows themselves. This substantially affects the application's transfer of control.
- Use -RUN for immediate execution; otherwise requests are 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 commands 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 command following the GOTO.

Multi-Input Windows

This is a *multi-input* window:



A multi-input window prompts you for information 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 Painter Screens.

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.

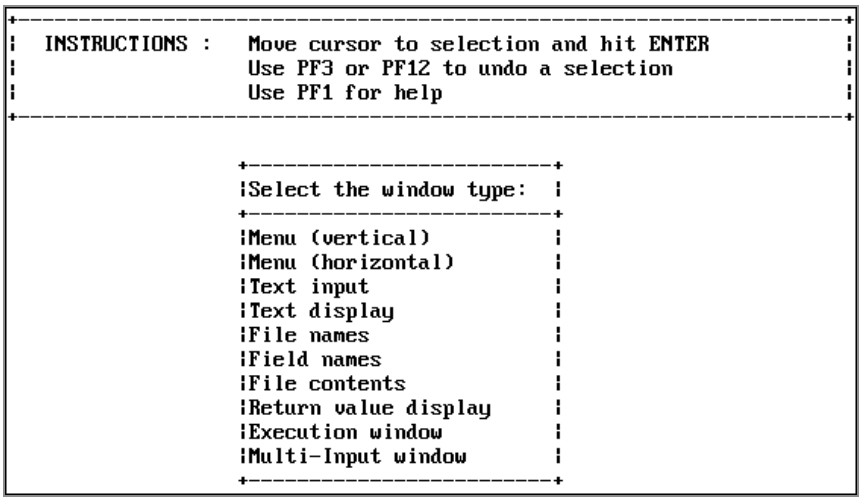
Use a unique name for each field on a multi-input window. To display the field names specified, use the Input Fields option on the Window Options menu.

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 describes how to create and implement text display window, vertical menu, and file names windows. This topic describes how to create horizontal menus (with or without associated pull-down menus) and multi-input windows.

Creating a Horizontal Menu

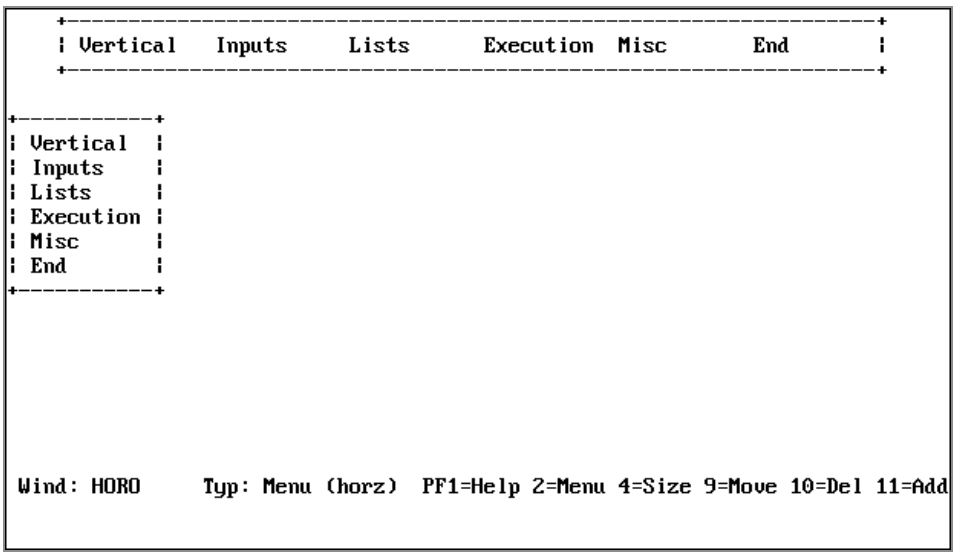
To create a horizontal menu, begin by placing the cursor at the **Menu (horizontal)** option on the Window Creation menu:



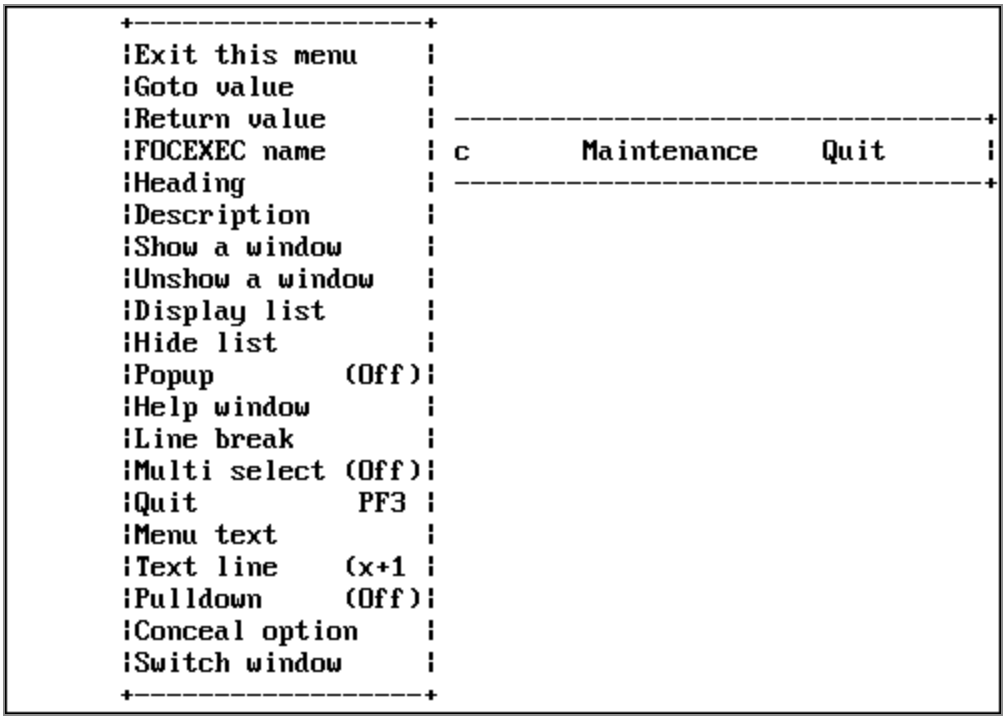
You are prompted to enter a name and brief description for the window, after which you reach the creation screen. On this screen:

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 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 **Enter** after each item; the item automatically appears on the horizontal menu.

The following is an example of a completed creation screen:



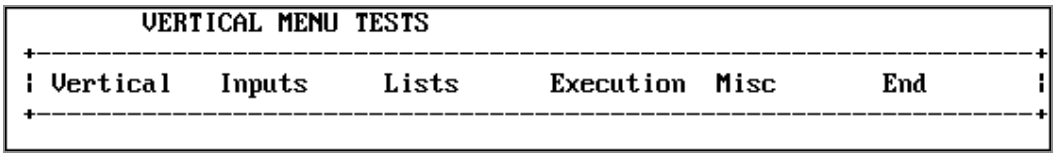
Once you have entered the items on your menu, there are several options you can select for each item. Move the cursor to any item and press **PF2** to display the Window Options menu:



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:



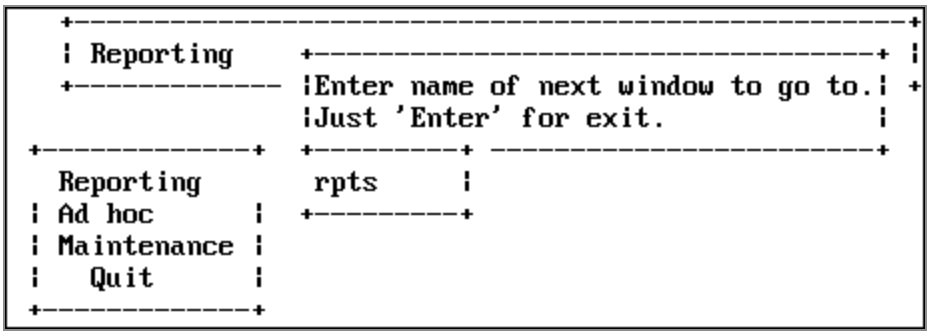
In this example, the Menu text VERTICAL MENU TESTS is positioned at Text line x-1, one line above the menu. To place the Text line two lines above the Menu text, change x-1 to x-2. For Text lines below the menu text, use x+1 or x+2.

You can also select the Pull-down option for a horizontal menu. With this option, you can assign a pull-down menu to be displayed for a horizontal menu item whenever the cursor is positioned on that item.

Pull-down Menus

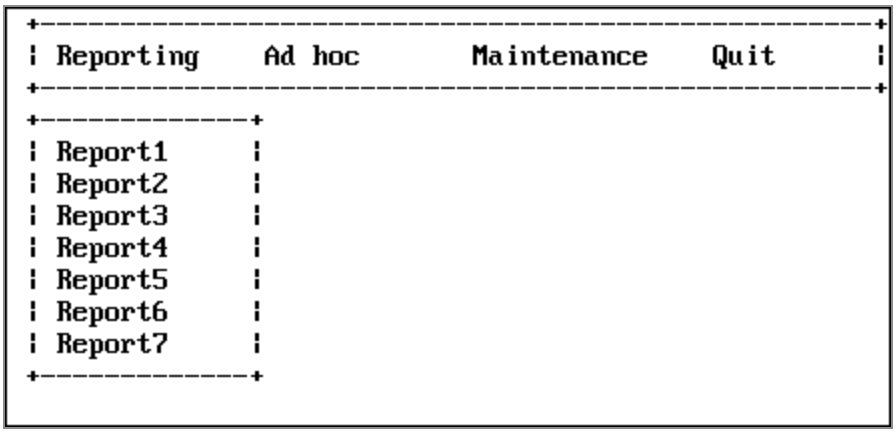
When you set the Pull-down option ON, you can display an associated pull-down menu for an item in a horizontal menu by positioning the cursor on that item. The default is OFF. To change the setting to ON, position the cursor on the Pull-down option and press **Enter**. Note that when Pull-down is set ON, Menu Text is automatically set OFF.

The associated pull-down menu must be a vertical menu. When creating the horizontal menu, you must assign a Goto value to point to the pull-down menu. To do so, position the cursor on the goto value, press **Enter**, and enter the name of the pull-down menu you want to display in the space provided:

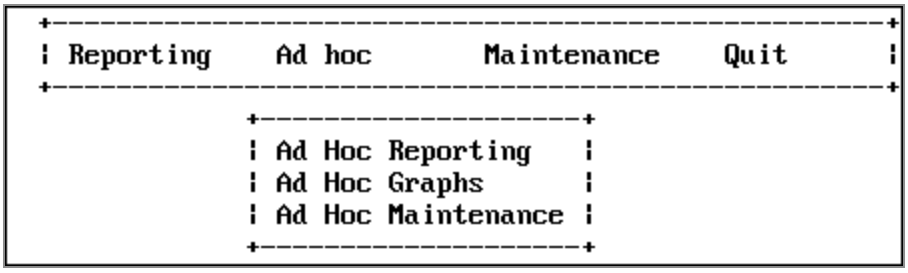


You must create the vertical menu, rpts, as you would any other vertical menu. See Tutorial: A Menu-Driven Application for examples.

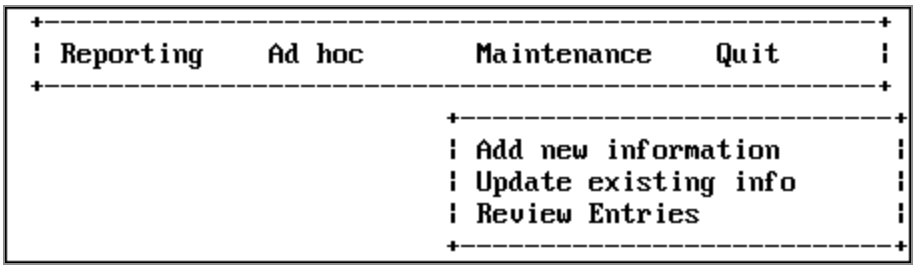
The following example shows a horizontal menu with the Reporting pull-down menu displayed:



The following screen shows the same menu with the Ad hoc pull-down menu displayed:



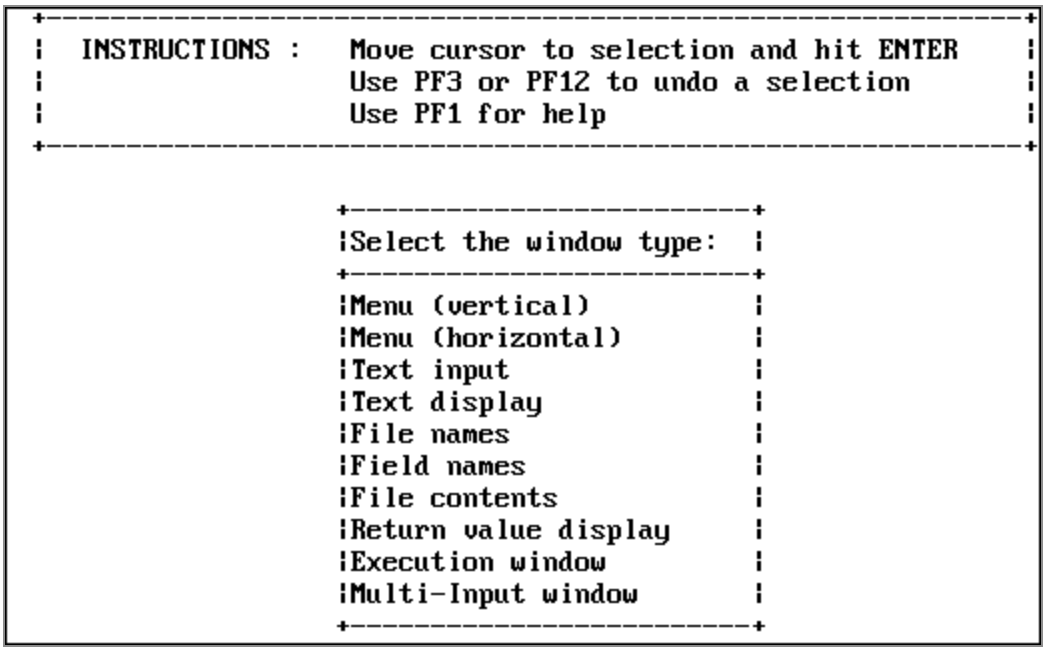
The following screen shows the same menu with the Maintenance pull-down menu displayed:



Note: To move from item to item in a horizontal menu, use **PF10** and **PF11**.

Creating a Multi-Input Window

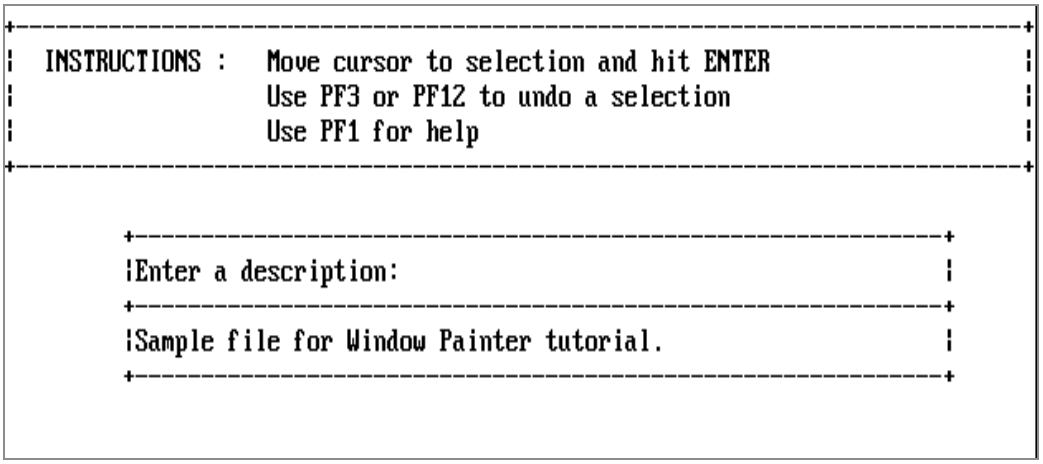
To create a multi-input window, begin by placing the cursor at the **Multi-Input window** option on the Window Creation menu and press **Enter**. You are then prompted for a name, description and heading. Place the window on the screen and size it as desired.



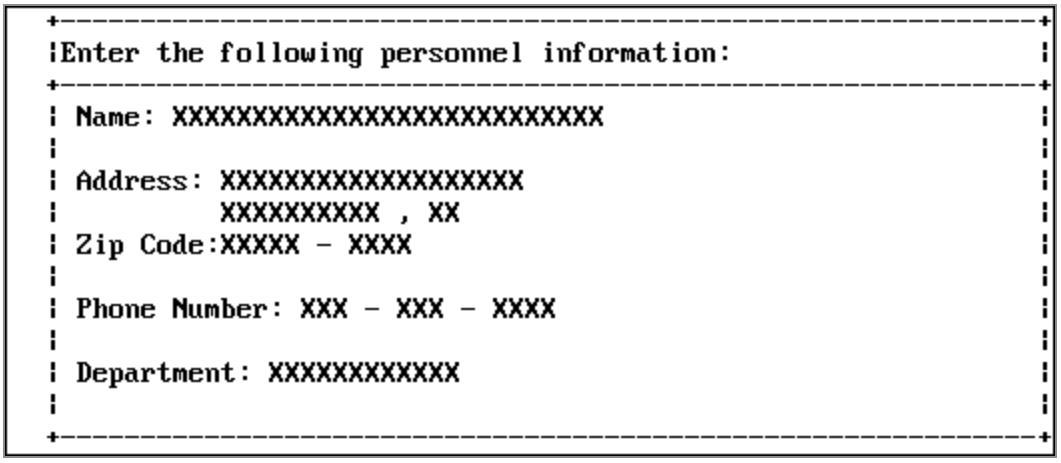
To place entries on the window:

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.

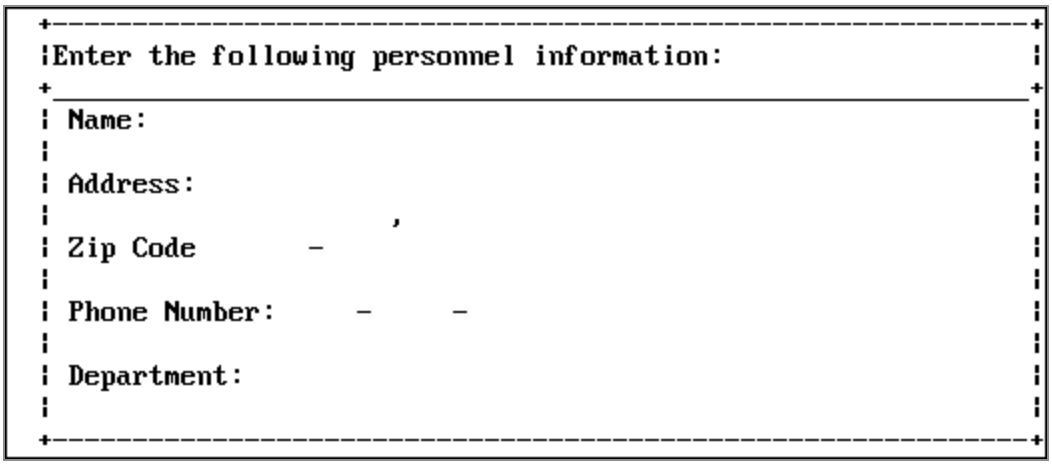


This is what the developer's screen looks like after several fields have been included in the multi-input window:



Note: Text fields may be supplied without headings or instructions. For example, see the city and state portion of the address line.

This is how the window appears when run as part of the application:



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                                                    ¦
¦&NUMB                                                        ¦
¦&DEPARTMENT                                                  ¦
¦&PFKEY        ENTR                                           ¦
¦&RETCODE             0                                       ¦
+-------------------------------------------------------------+
```

Note: To move from field to field in a multi-input window, use the Tab key.

Integrating Windows and the FOCEXEC

The windows created with Window Painter are designed for use within an application FOCEXEC. This topic discusses how to integrate the windows into your FOCEXEC.

Invoke the Window Facility

To invoke the Window facility, insert the following Dialogue Manager command in your FOCEXEC

```
-WINDOW windowfile windowname [PFKEY|NOPFKEY] [GETHOLD] [BLANK|NOBLANK]
[CLEAR|NOCLEAR]
```

where:

windowfile

Identifies the file in which the windows are stored. This is a member name. The member must belong to a PDS allocated to ddname FMU.

windowname

Identifies which window in the file to display first. Can be set in Window Painter or in first window displayed. This is optional.

PFKEY

Enables testing for function key values during window execution.

NOPFKEY

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 command is encountered. This is the default setting.

NOBLANK

No amper variable values are cleared when the -WINDOW command 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 is 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 that use more than one -WINDOW command.

Transferring Control in Window Applications

When the -WINDOW command 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 Integrating Windows and the FOCEXEC.)

Once control passes back to the FOCEXEC, control only returns to the Window facility if another WINDOW command is encountered.

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 commands and windows are executed).

```
1. -START
2. -WINDOW REPORT R1 PFKEY
   -*
3. -*Control is transferred from the above command
   -*to window R1 in window file REPORT.
   -*
4. -IF &PFKEY EQ PF05 GOTO LABEL1;
   -*
   -*Control returns to the above command from
   -*window R2 in window file REPORT.
   .
   .
   -LABEL1
5. -WINDOW REPORT R3
   -*
6. -*Control is transferred from the above command
   -*to window R3 in window file REPORT.
   -*
7. -IF &R3 EQ EXIT GOTO EXIT;
   -*
   -*Control returns to the above command from
```

```
-*WINDOW R3 in window file REPORT.
.
.
-EXIT
```

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 command 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 command following the -WINDOW command (Step 7).

The flow of control has certain implications for the design of your window applications:

- Any time you 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 Integrating Windows and the FOCEXEC).
- 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 pass control from the FOCEXEC to the Window facility you must insert the -WINDOW command 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 any window.
- 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 command should follow the -WINDOW command, and a second -WINDOW command should be placed after the -IF command to transfer control back to the window file.
- If you want to clear an existing set of variable values, return control to the FOCEXEC and execute another -WINDOW command with the BLANK option in effect.

To back up a step during window execution, the user may press **PF12** or **PF24**. This does 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 Integrating Windows and the FOCEXEC.

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 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 menu and horizontal menu windows, you assign any return value 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 window return value is collected as soon as control passes to the window, without the user selecting 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 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.

Return Value in a Menu-Driven Application

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 that meet certain file-identification criteria 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 creating your windows, you 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 menu 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 Integrating Windows and the FOCEXEC, 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 command. The goto value points your application to a new window in the window file; the -GOTO command 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 returns to the previous window upon completion of the current window, you must 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 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 contains 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

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 command, to explicitly set those actions.

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 command 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
-RUN
```

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 Customizing Your Environment, 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 Testing and Debugging With 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 commands 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 immediately returns control to the FOCEXEC if that key was set to RETURN (or to HX, CANCEL, or END).

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 is not 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 ibi FOCUS Prompt

You can execute a window directly from the FOCUS command prompt.

Execute a Window From the ibi FOCUS Prompt

```
EX 'windowfile FMU' [windowname] [PFKEY|NOPFKEY] [BLANK|NOBLANK]
[CLEAR|NOCLEAR]
```

where:

windowfile

Is the file containing the windows. It must have ddname FMU, and appear within single quotation marks.

windowname

Identifies the first window to be executed. If a window name is not specified, FOCUS executes the default start window, or the first window created.

PFKEY

Tells FOCUS you will test for function key values during execution.

NOPFKEY

Tells FOCUS you will not test 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.

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 is shown an automatically generated list of FOCEXECs from which to pick. A user who chooses to create a new request is 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 is 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 FOCEXEC.

```
-START
-WINDOW SAMPLE MAIN
-*
-*Control is transferred from the above command
-*to window MAIN in window file SAMPLE.
-*
-IF &MAIN ...
-*
-*Control returns to the above command
-*from option "Exit?" in window MAIN,
-*from option "New Request?" in window EXECTYPE,
-*and from every selection in window EXECNAME.
-*
.
.
.
-GOTO START
-EXIT
```

Window	If option selected is...	Then go to:
MAIN	Report? Graph?Exit?	window EXECTYPEwindow EXECTYPEback to FOCEXEC
EXECTYPE	Existing Request?New Request?	window EXECNAMEback to FOCEXEC
EXECNAME	The options in this window are a list of report and graph requests from which the user can select.	Control is transferred back to the FOCEXEC.

Creating the Application FOCEXEC

A FOCEXEC called SAMPLE drives this application.

Begin by using the TED editor to create the FOCEXEC file SAMPLE. At the FOCUS prompt, type

```
TED 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 commands, and must begin in the first column. Dialogue Manager is discussed in Managing Flow of Control in an Application.

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 command.

```
-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 command transfers control to the Window facility. SAMPLE is the name of the window file this application uses and we will create it in this tutorial. MAIN is the window where the procedure begins.

Control does 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 FOCEXE branches to -GENERATE; if XXIT, to -EXIT. Each return value corresponds to a selection on the menu window MAIN.

3. This label beings to GENERATE section of the FOCEXEC
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 that executes 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----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 command 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 commands returns to -START, as in note 10.
16. This command ends FOCEXEC execution.

Creating the Window File

The -WINDOW command 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:

BORDER	A text display window used as a background display for the other windows.
BANNER	A text display window that introduces the application.
MAIN	A vertical menu from which the user can choose to create a graph or a report, or exit the application.
EXECTYPE	A vertical menu from which the user chooses to execute an existing procedure or create a new one.
EXECNAME	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.

All these windows are included in the window file named SAMPLE. Start by building that window file.

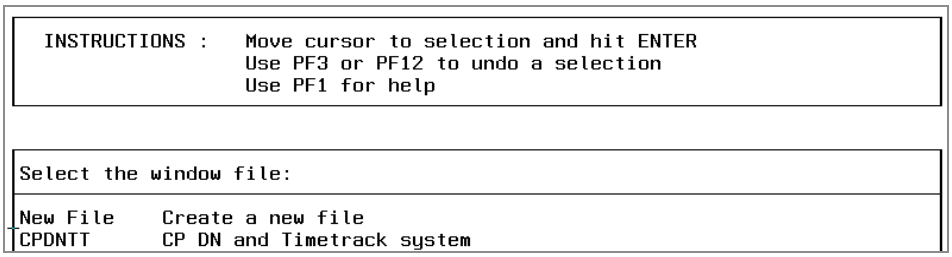
Before you can use Window Painter to create a window file, a PDS must be allocated with ddname FMU, LRECL 4096, and RECFM F. BLKSIZE 4096 is recommended.

You can reach the FOCUS Window Painter Entry Menu by typing

```
WINDOW [PAINT]
```

at the FOCUS prompt, and pressing **Enter**.

The Entry Menu is the first screen you see:

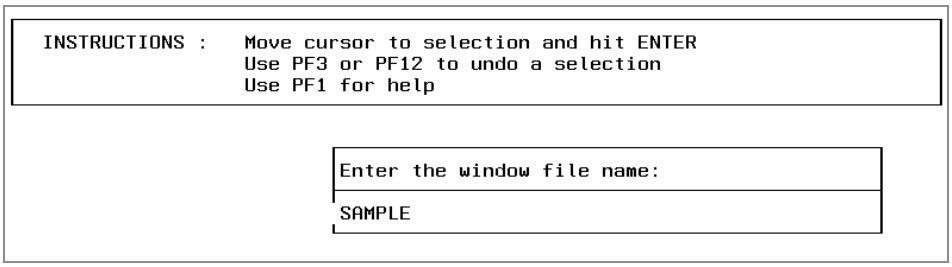


Since you are creating a new window file, choose NEW FILE, and press **Enter**. The next screen you see prompts you to name the window file.

Since the FOCEXEC looks for a window file named SAMPLE, type

```
SAMPLE
```

and press **Enter**.

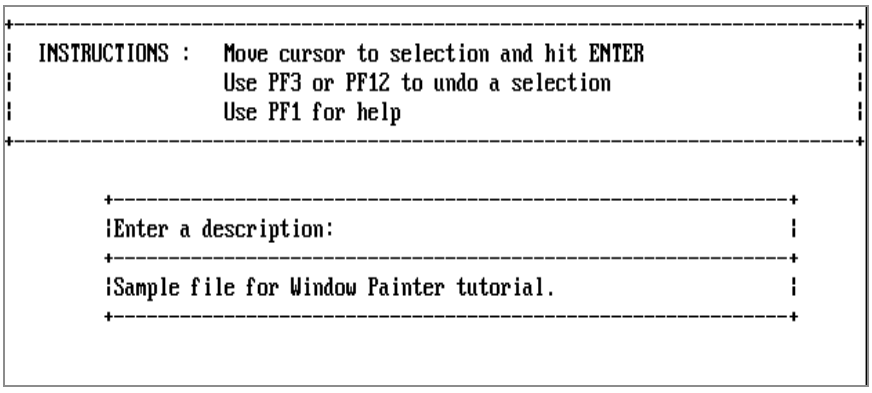


A screen appears asking for a description of the window file.

Type

```
Sample file for Window Painter tutorial
```

and press **Enter**.

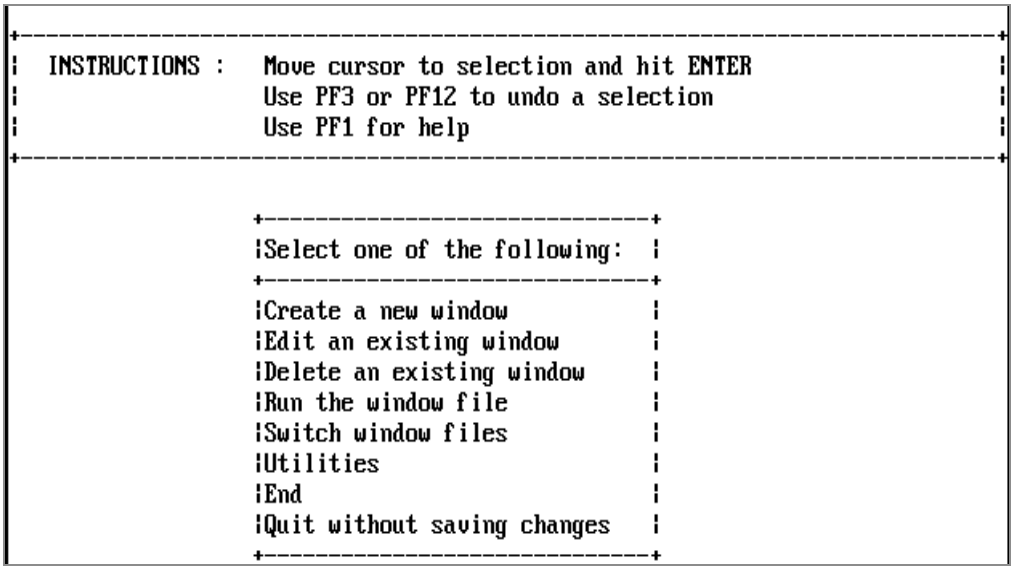


Creating the Text Display Window Named BORDER

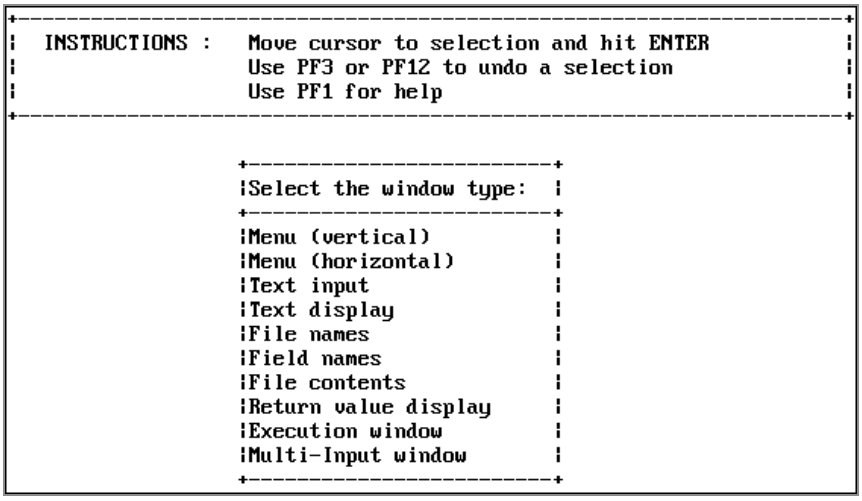
Now you are ready to create the first window. The Window Painter Main Menu screen appears. Select

```
Create a new window
```

and press **Enter**.



The Window Creation Menu asks what kind of window you want to create.



The BORDER window is the first window you create for the application. BORDER supplies a background border for other windows. It is a text display window, so select

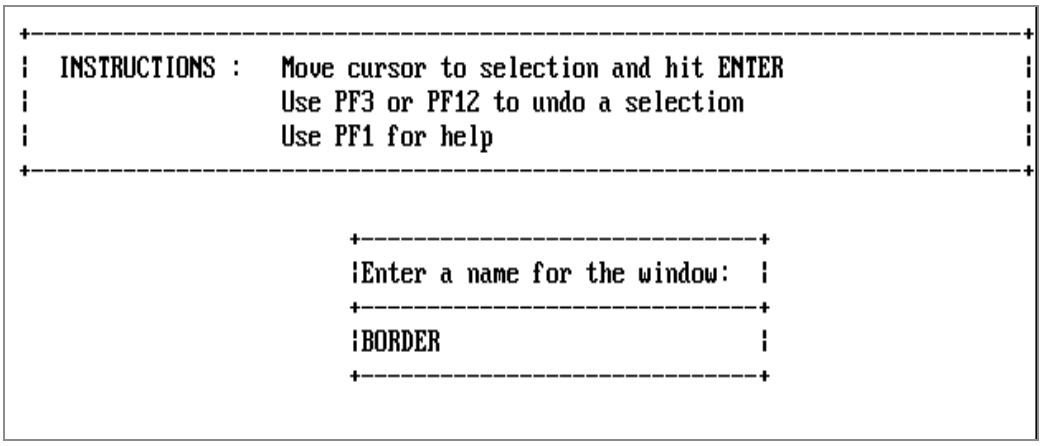
```
Text display
```

and press **Enter**.

Next, you are asked to name the window. Type

```
BORDER
```

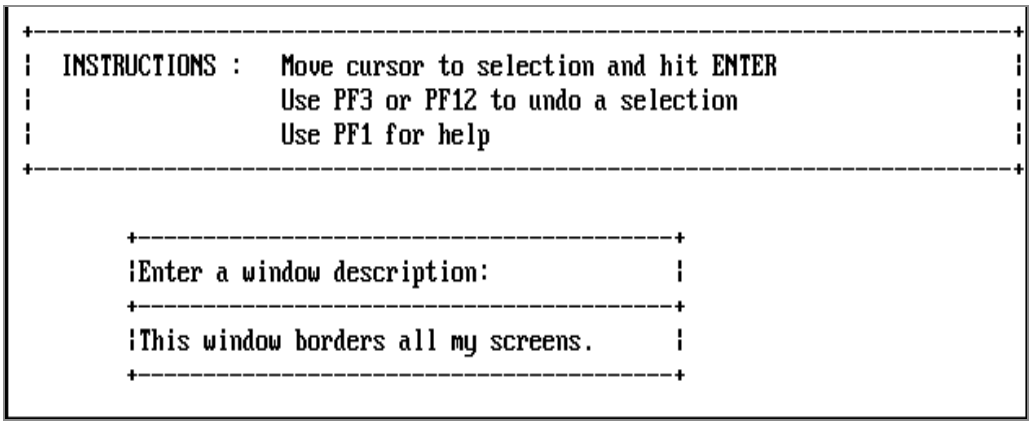
and press **Enter**.



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.

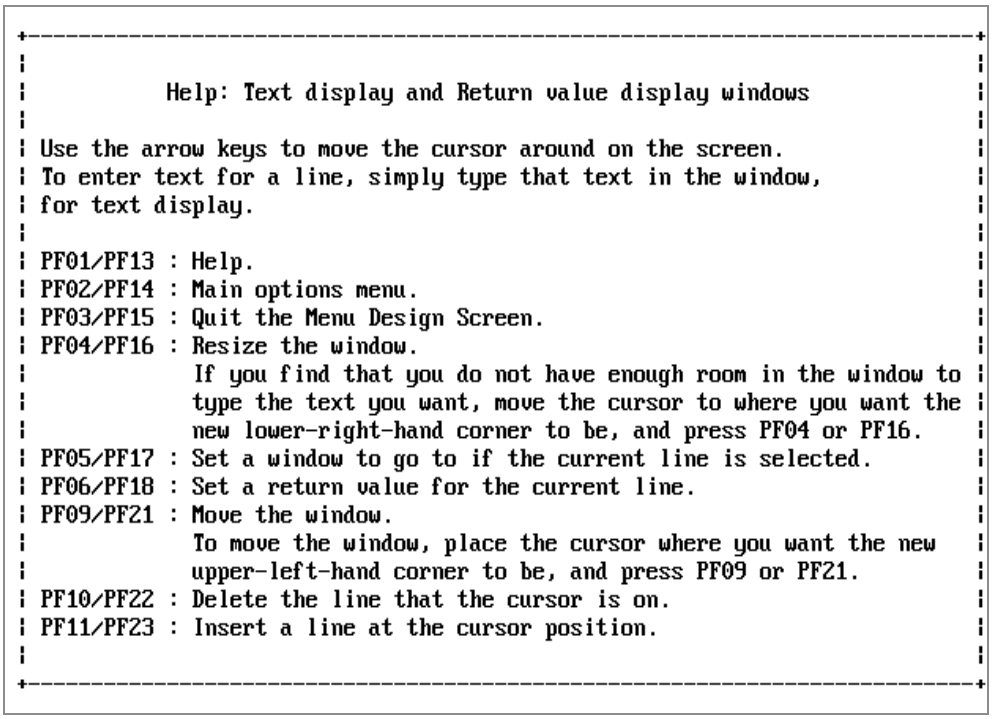
The Window Design Screen displayed now is nearly blank, with a cursor for you to position where you want the upper left-hand corner of BORDER to be. Leave the cursor where it is and press **Enter**.

A small box appears around the cursor: this is the window. 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 is 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 that can contain data or text: this is the row immediately above the bottom border.

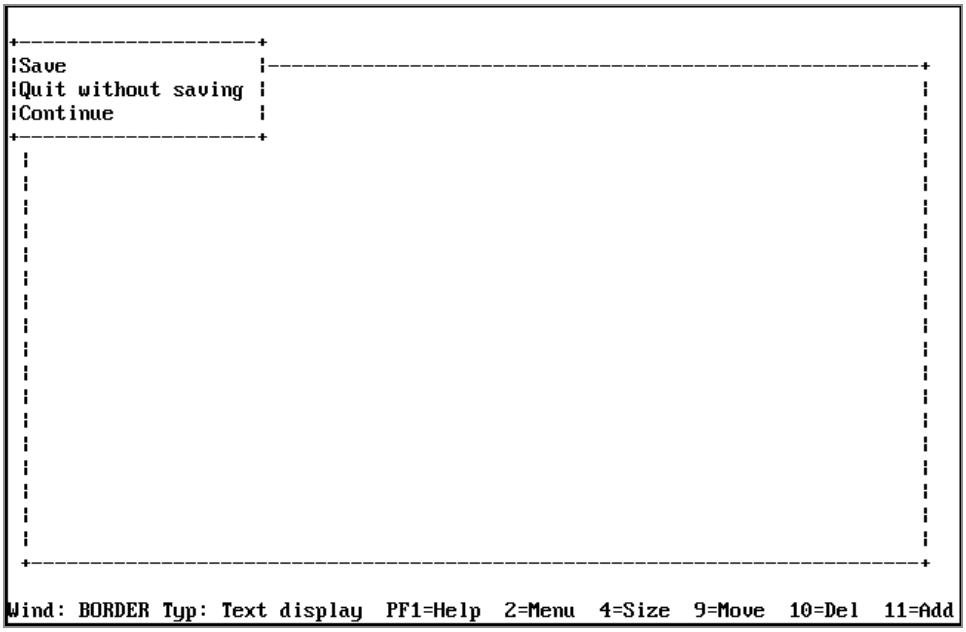
This window's border forms the background border for the other windows in this application.

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:



Press **Enter** to continue.

Now that the window is complete. Press **PF3** and save the window.



Press **Enter** to select Save. You return 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.

Enter text to be displayed in the window. Reposition the cursor on the first line within the window, 10 spaces to the right of the window's left border, and type:

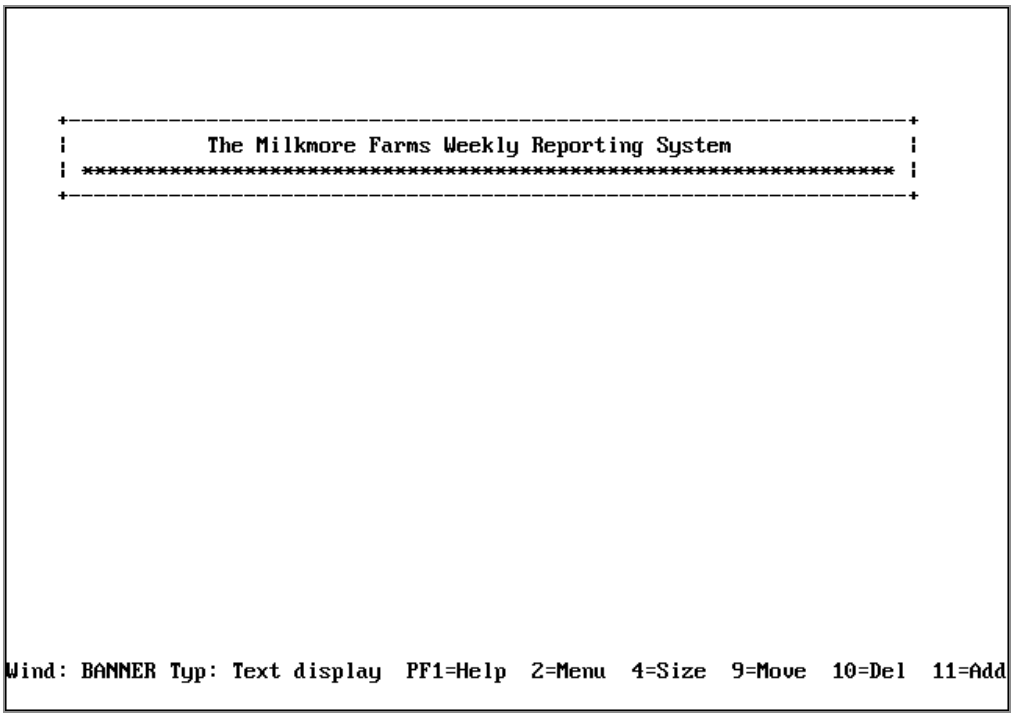
```
The Milkmore Farms Weekly Reporting System
```

Type a line of asterisks (*) across the window's second line. (Begin at the second column within the window, because the first column of every window is protected.)

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:



Press **PF3** and save the window.

Creating the Vertical Menu Window Named MAIN

You will now 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**.

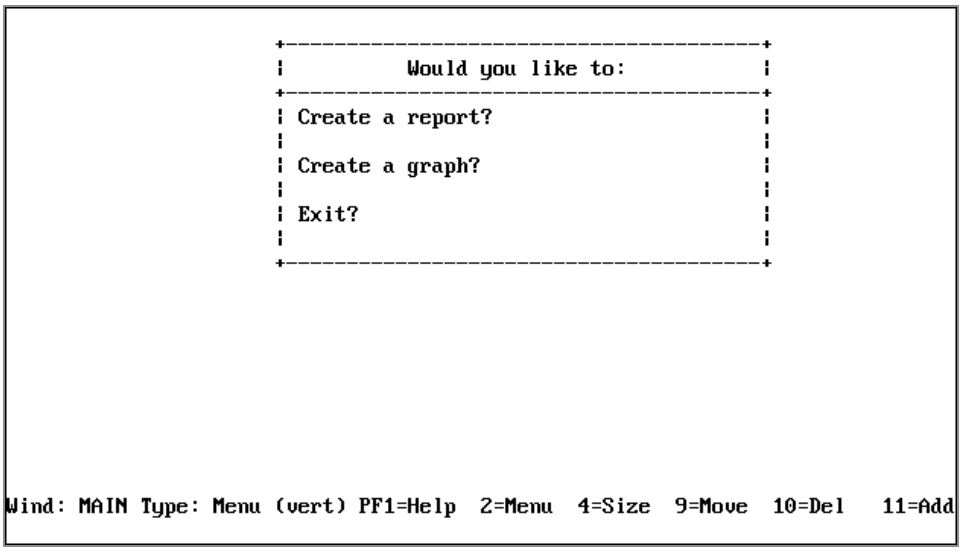
When prompted for a heading, type 10 spaces, then

```
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 is created wide enough to contain the heading. Now position the cursor six rows below the window's bottom edge, and 10 columns to the right of its right edge. Press **PF4** and the window is resized.

Type the following menu options as they appear below:

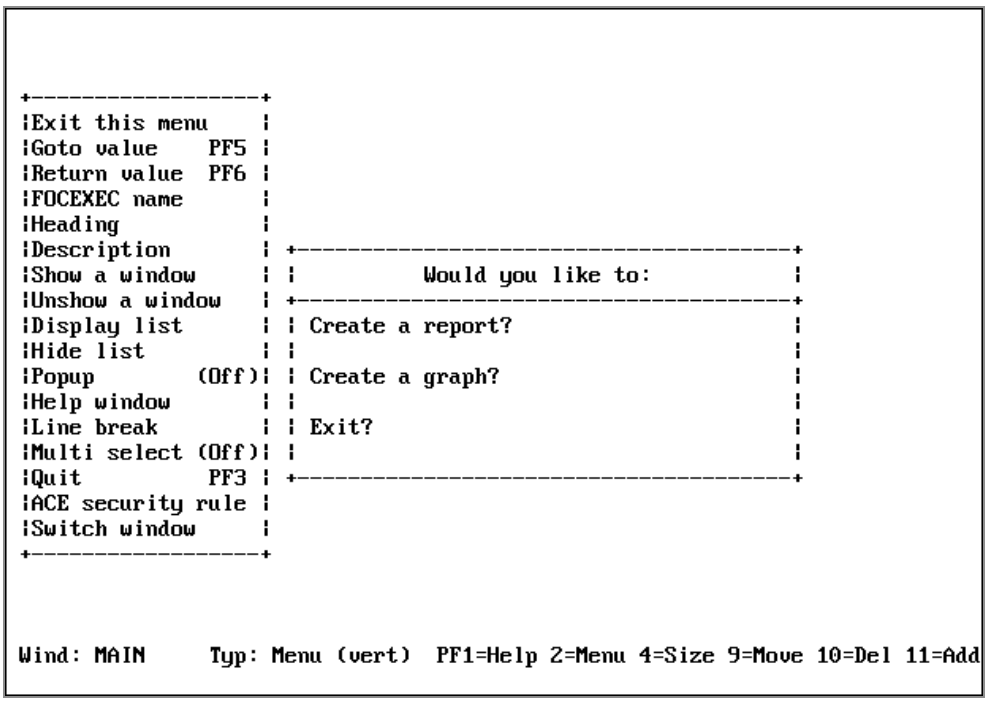


You assign goto and return values for each menu option. To assign either value to an option, the cursor must first be on that option.

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 is prompted to either execute an existing report or create a new one. The window that displays the prompt is called EXECTYPE, so the goto value of the first two menu options is EXECTYPE.

Move the cursor to

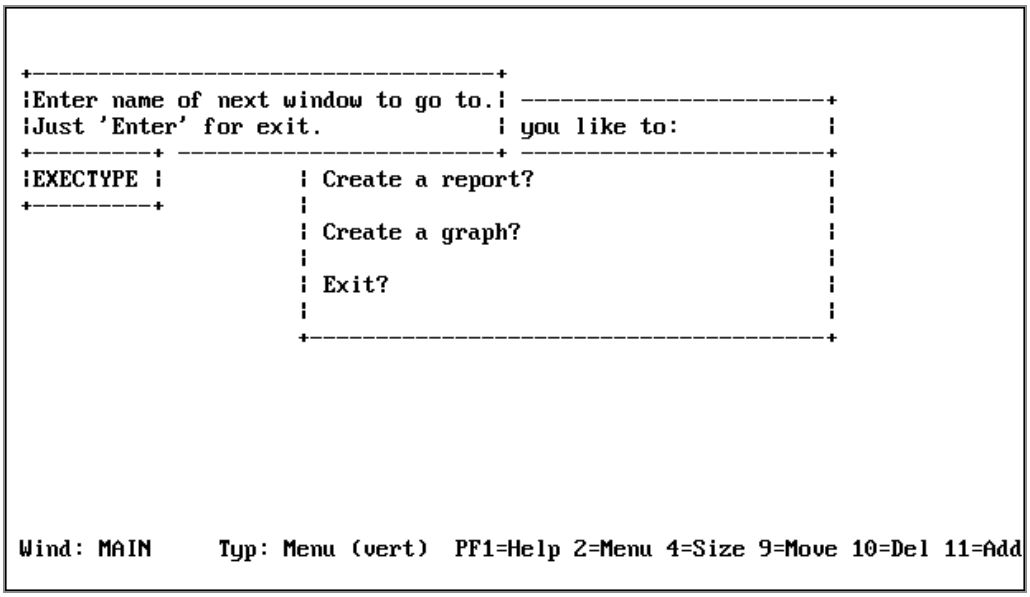
```
Goto value
```

and press **Enter**.

In the space provided, type

```
EXECTYPE
```

and press **Enter**.



The return value collected by this window—&MAIN—is 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

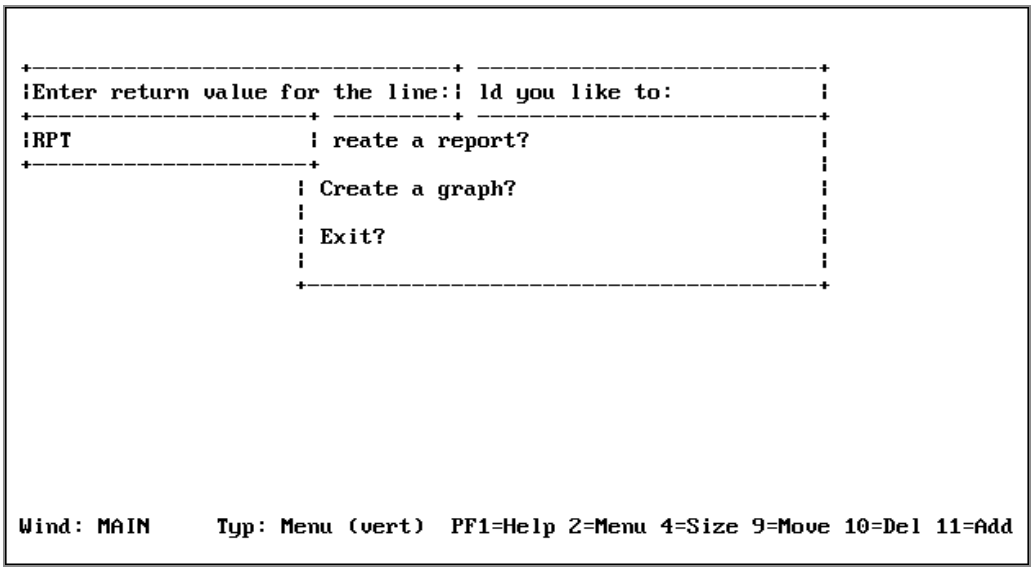
```
Return value
```

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**.

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:

```
GRPH
```

Leave the Window Options menu and move the cursor to

```
EXIT?
```

For this option, you do not assign a goto value. Since it exits to the FOCEXEC, there is no other window to be displayed.

Repeat the steps to assign the return value:

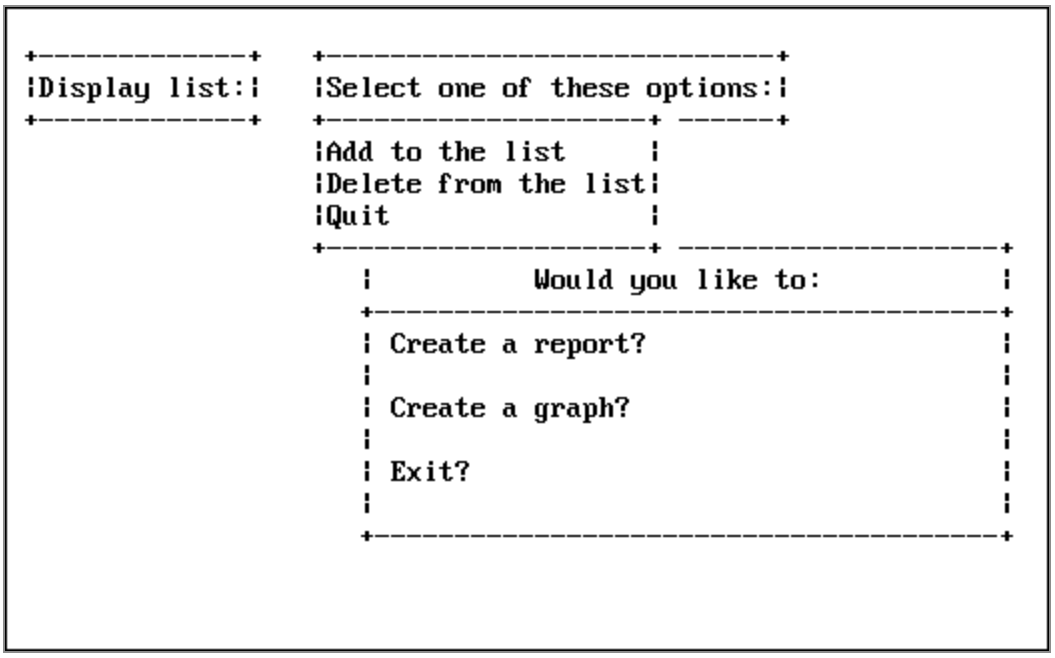
```
XXIT
```

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 each 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 are 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 is 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

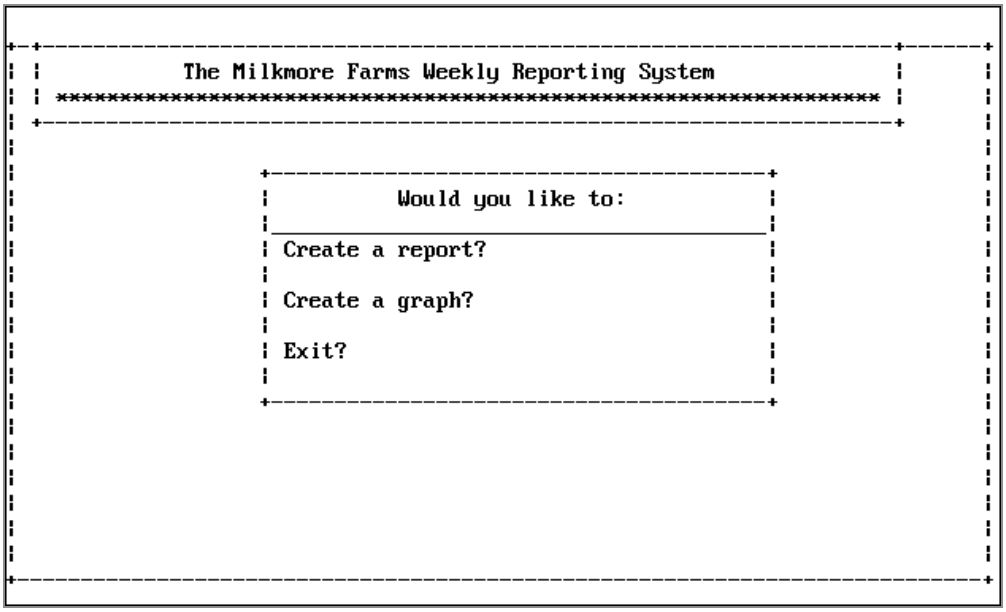
```
Run the window file
```

and press **Enter**.

Select

```
MAIN
```

as the starting screen. Press **Enter**, and the following appears:



Position the cursor on the "Create a report" line. When you press **Enter** to continue the display, you see an error message because EXECTYPE—the goto value—has not been created yet. Ignore it, and press **Enter** to continue. You see a screen displaying amper variables for this window and the 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 create is also 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.

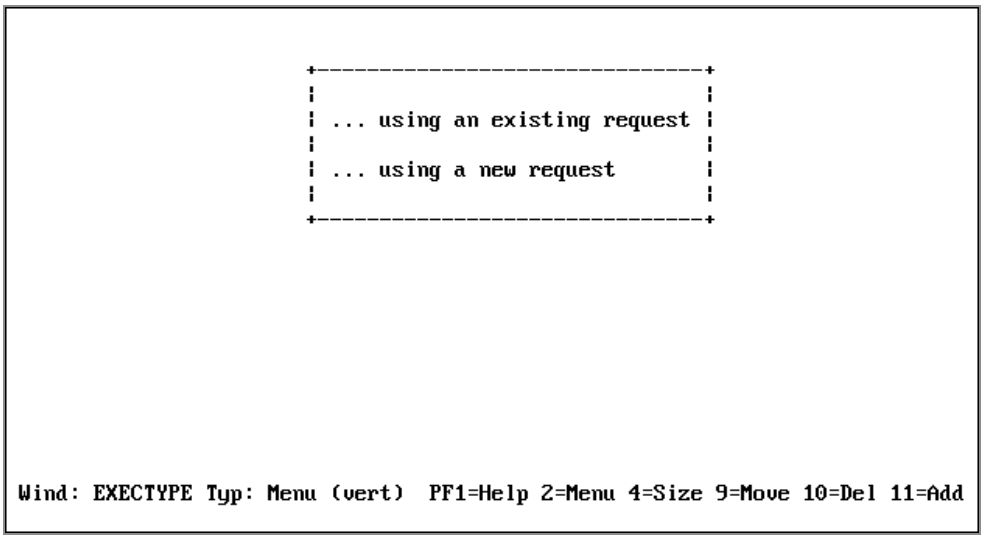
When prompted for a description, type

```
Create a new FOCEXEC 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 Return value, a GOTO value, and a FOCEXEC name. When prompted for the Return value, enter EXIST and press **PF5**. You are prompted for A GOTO value. Press **Enter**, and you are prompted for a FOCEXEC name. Press **Enter**.

If you select

```
... using an existing request.
```

from the EXECTYPE menu, the file names window EXECNAME displays next. EXECNAME contains a list of existing FOCEXEC files from which you may choose.

Move the cursor to the second menu item.

Consider the return and goto values for this option.

If you choose to create a new report or graph request, EXECNAME is not displayed. Rather, control must pass back to the FOCEXEC, which executes 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
```

For control to pass to the FOCEXEC if this option is chosen, do 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 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**. (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 that 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.
```

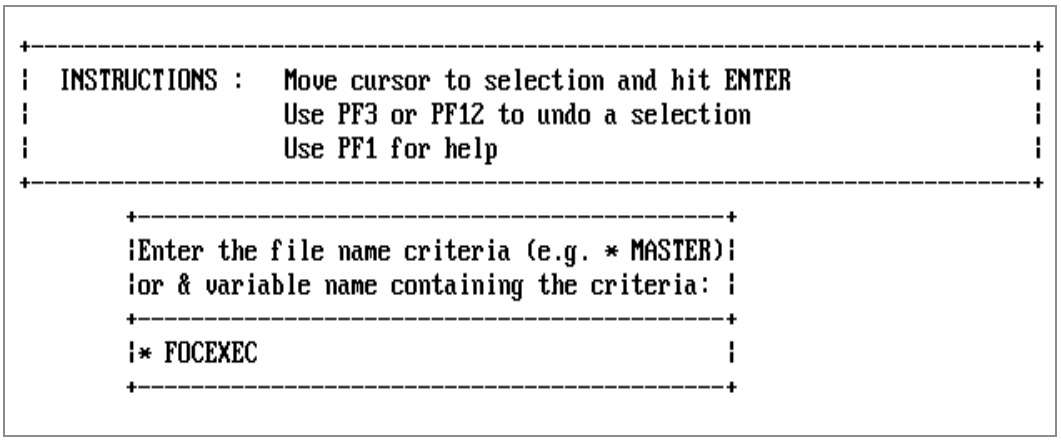
Enter an explanatory heading:

```
Select the request you want to execute and press ENTER:
```

You are prompted for file-identification criteria. Type

```
* FOCEXEC
```

and press **Enter**.



When the application is executed, this selects 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 10 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 is automatically 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 is plugged into the line

```
EX &EXECNAME
```

and the report or graph request is executed.

In order for this to happen, you must return control to the FOCEXEC assigning no goto value to this window.

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 actual return values set when the window is created or edited, the "return value" that 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 executes 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**.

Invoke Window Painter

```
WINDOW [PAINT [filename]]
```

where:

PAINT

Is optional.

filename

Is the name of the window file that you want to work with. This is a member name. The member must belong to ddname FMU.

If you do not specify file name, you begin your Window Painter session at the Entry Menu where you can choose a window file to use or create a new window file. If you do specify file name, you 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 are asked if you want to create a new file. If not, the Window Painter Entry Menu is displayed.

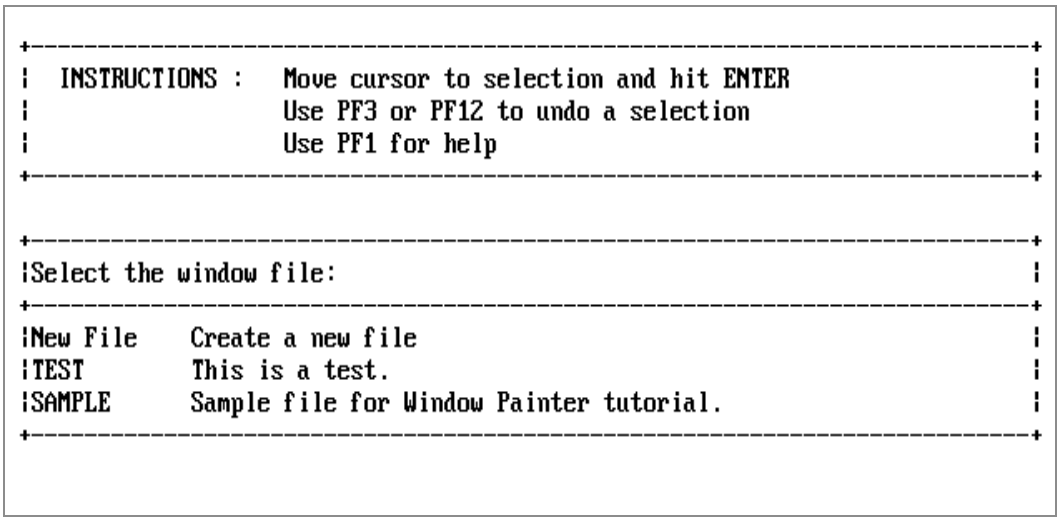
Entry Menu

You can reach the Window Painter Entry Menu by typing

```
WINDOW [PAINT]
```

at the FOCUS prompt, and pressing **Enter**.

The Entry Menu is the first screen you see:

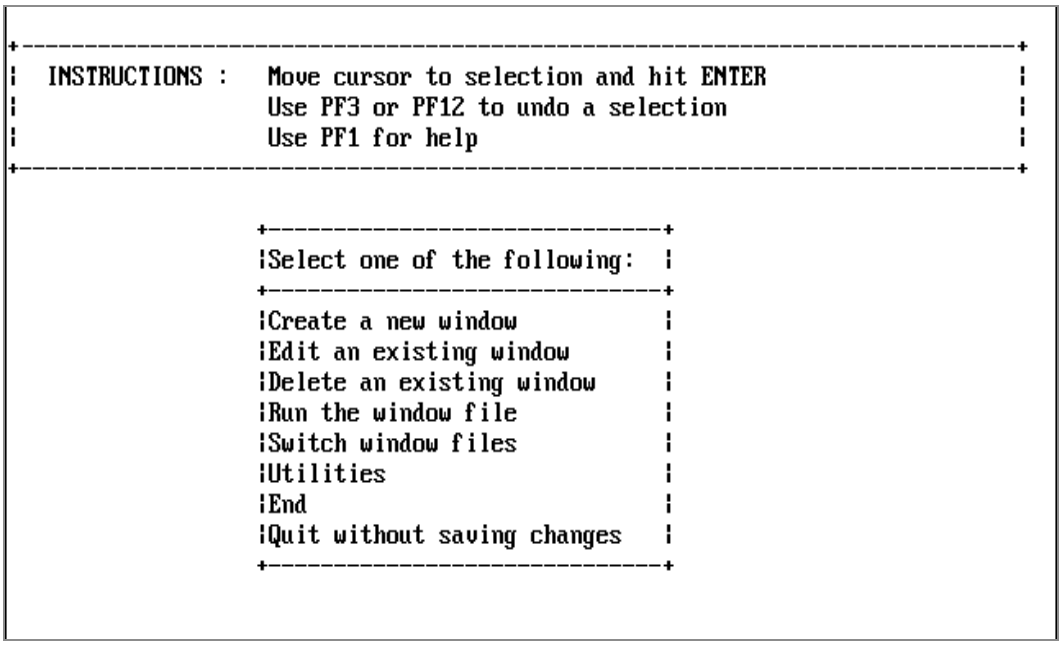


The Entry Menu invites you to choose a window file in which to work. If you are creating windows for a new application, you should start a new window file. If you are maintaining or creating windows for an existing application, use the window file that corresponds to your application.

When you become comfortable working with windows, you can write FOCEXECs that include branching between window files. Refer to Integrating Windows and the FOCEXEC for a detailed discussion on branching and transferring control.

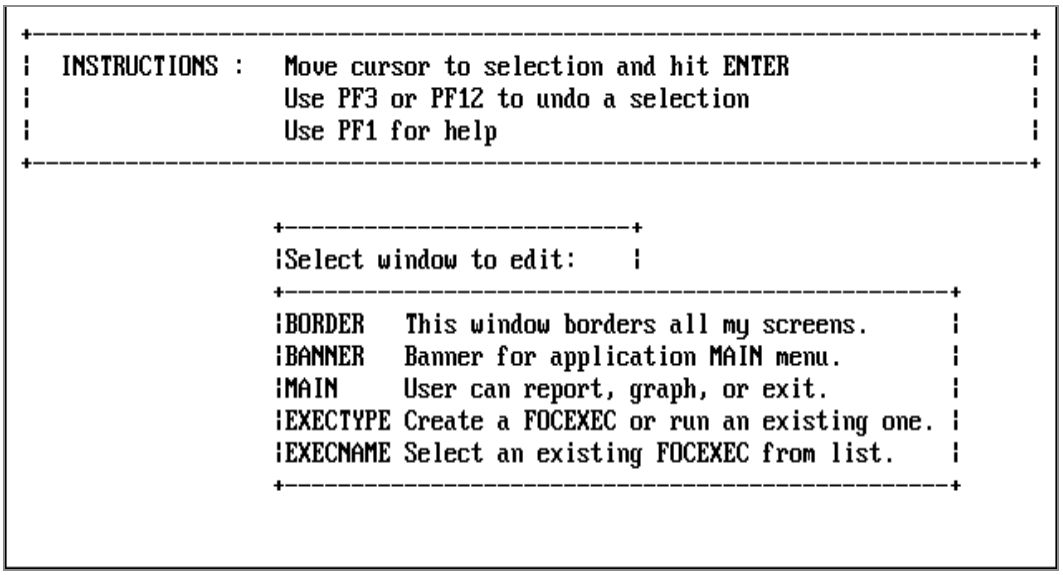
Main Menu

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:

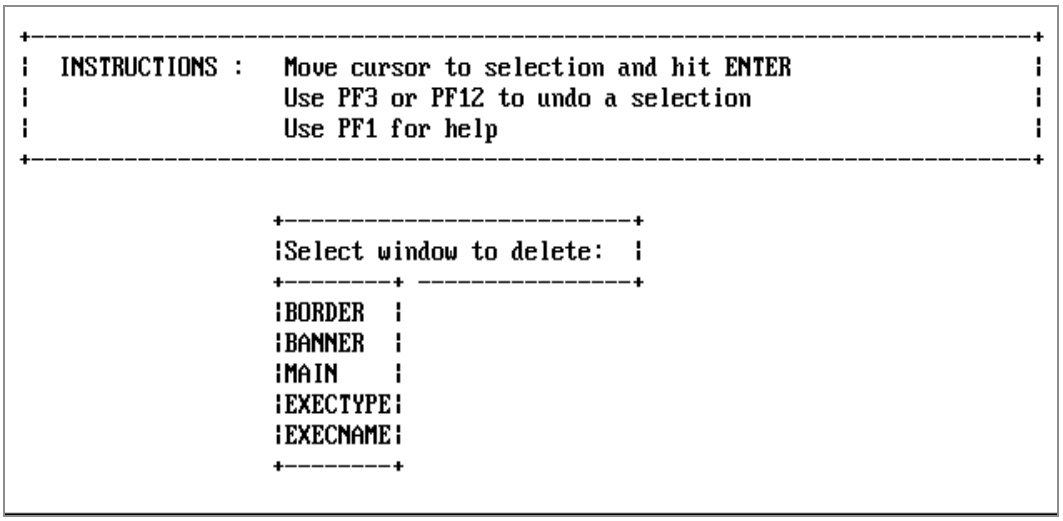


The following table summarizes the options on the Main Menu, along with illustrations of screens that appear when you select the options:

Menu Option	Description
Create a new window	Brings up the Window Creation Menu. You can select the type of window to create.
Edit an existing window	Brings up a list of windows in your current window file. You can select the one to edit.



Menu Option	Description
Delete an existing window	Brings up a list of windows in your current window file. You can select the one to delete.



Menu Option	Description
Run the window file	Brings up a list of windows in your current window file. You

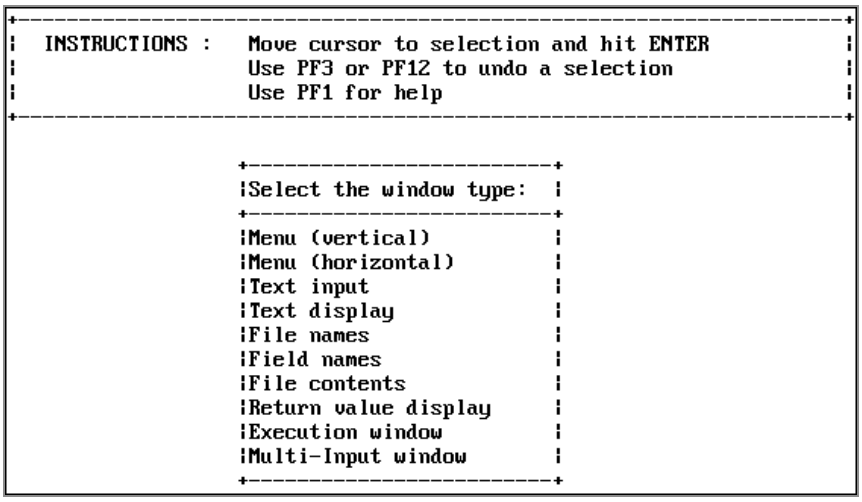
Menu Option	Description
	can select the one from which to start running the window file. After the window file is run, the windows' amper variable values are displayed. The display includes the first 20 characters of each value. This option shows you how your windows work without executing the FOCEXEC. Use this option to test your window file.
Switch Window files	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.
Utilities	Brings up the Utilities Menu, which is discussed in Window Painter Screens.
End	Returns you to native FOCUS. All work saved during the Window Painter session is kept.
Quit without saving	Returns you to native FOCUS. All work saved during the Window Painter session is discarded.

Window Creation Menu

You can reach the Window Creation Menu by selecting

```
Create a New Window
```

from the Main Menu. The following screen appears:



You need to select the type of window to create. You are asked to enter an 8-character name and an optional 40-character description. These are for your use only and 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 indicating that any character or sequence of characters can occupy that position. 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 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 commands or TABLE requests. You are prompted for the window name and heading. Once a window has been specified, the Window Design screen opens.

For complete information about the types of windows you can create in Window Painter, see Window Files and Windows.

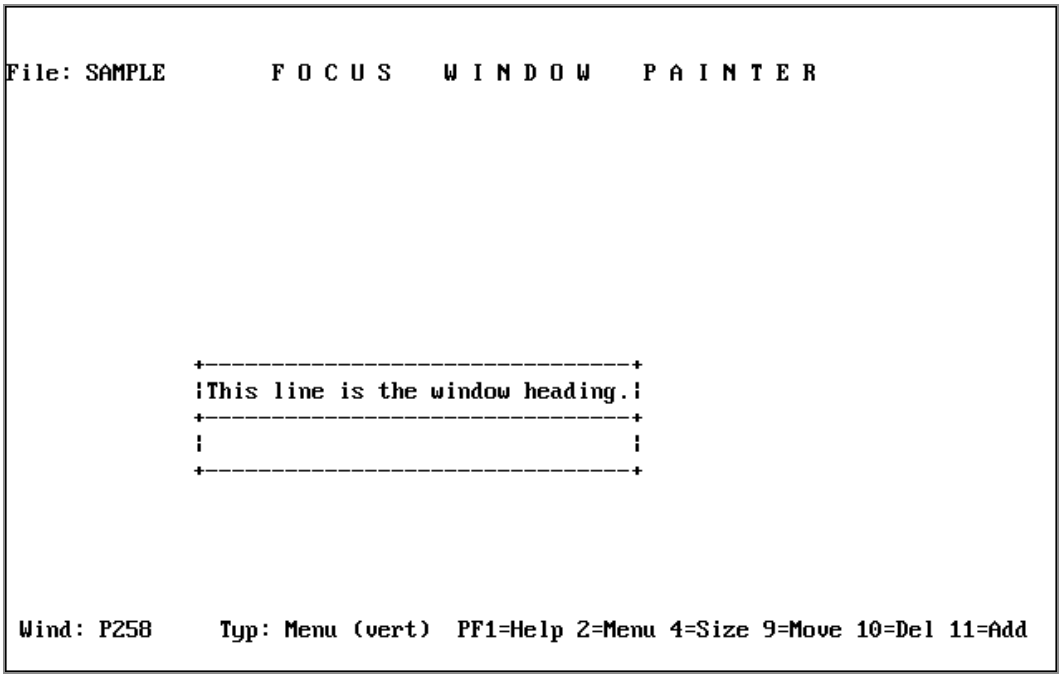
The next screen displayed is the Window Design Screen, discussed in the next section. This screen enables you to enter information, and position and size your window.

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**.

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 allows you to use the keyboard to manipulate the window you are creating.

The following chart summarizes Window Design Screen key functions in all window types.

PF Key	Function
PF1	Displays a window of help information.
PF2	Displays the Window Options menu. This menu is discussed in Window Painter Screens.
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 Return value, the GOTO value, and the FOCEXEC name for the active window.
PF6	Sets the return value of the line that 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.

PF Key	Function
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, the window itself is 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, the window itself increases by one line.
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

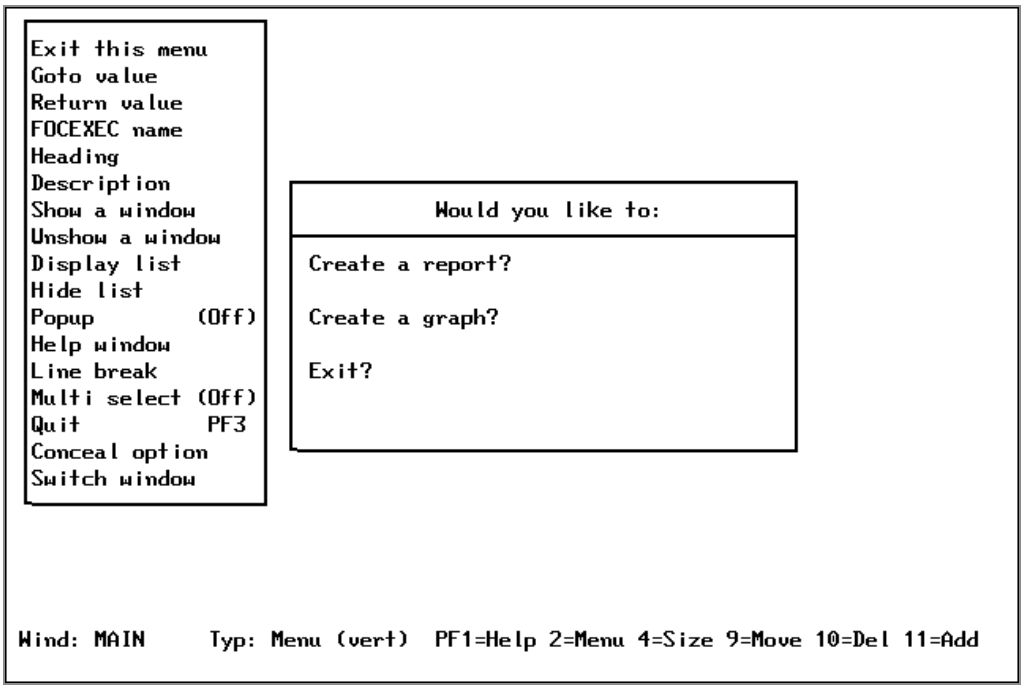
```
(MORE)
```

is displayed on that border to remind you of more lines of contents hidden beyond that border. You can view these lines by scrolling 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**. 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

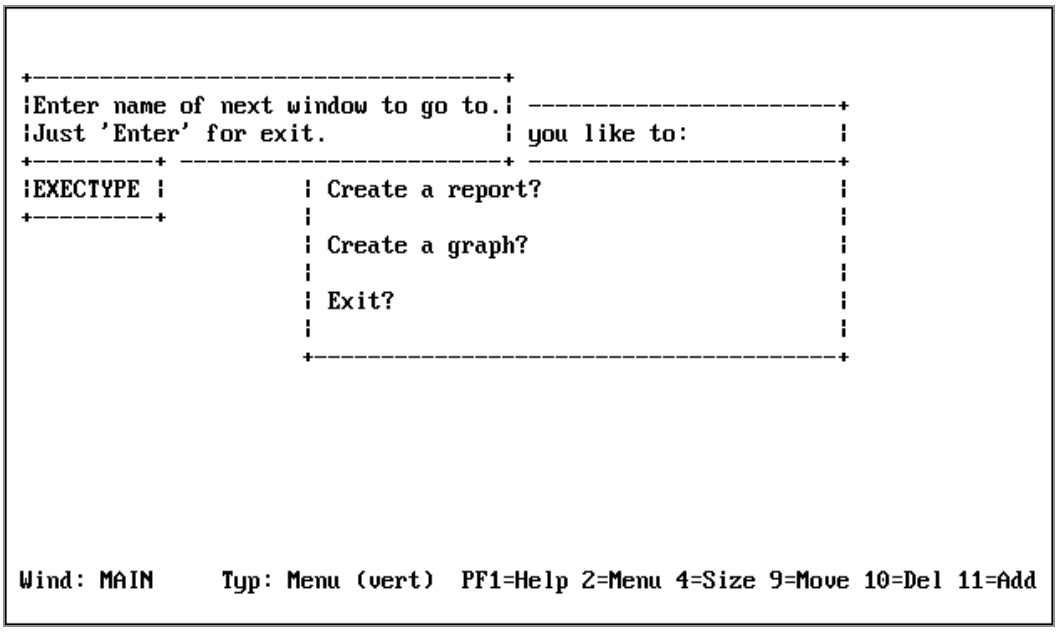
When the Window Design Screen is displayed, pressing **PF2** brings up the following Window Options Menu:



The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

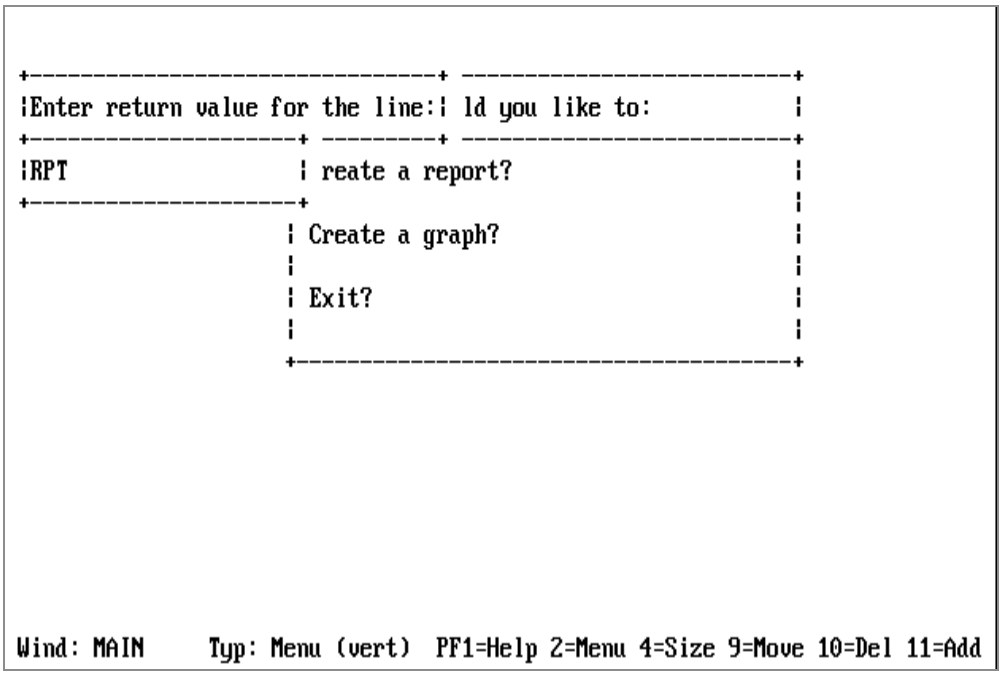
Menu Option	Description
Goto value	Selecting this option allows you to specify the next window in the path from this selection field or window. You are asked to supply the name of the window. (It does not matter whether or not this window exists. You can create it later, but remember the name chosen.) 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

Menu Option	Description
	up. Select Goto value from the Window Options Menu. You are 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.



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, where the 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.

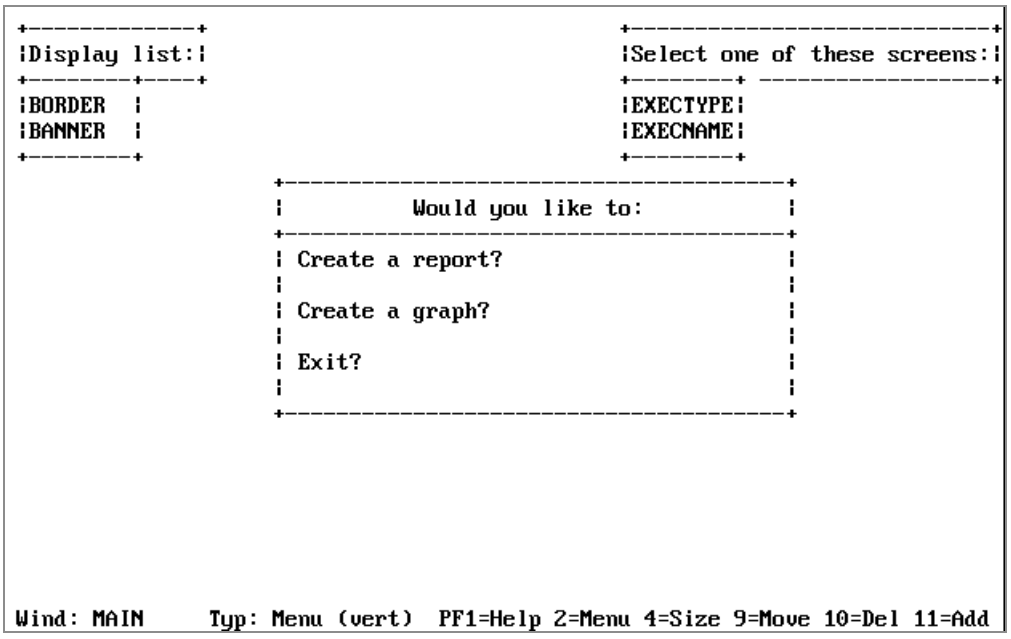
Menu Option	Description
	To assign a return value, your cursor must be on the proper line. Select Return value from the Window Options Menu and you are prompted to enter a return value. Note that for file names, field names, and file contents windows, the value that you enter is the file-identification criterion for that window. Type the value in the space provided and press **Enter** again to assign the return value .



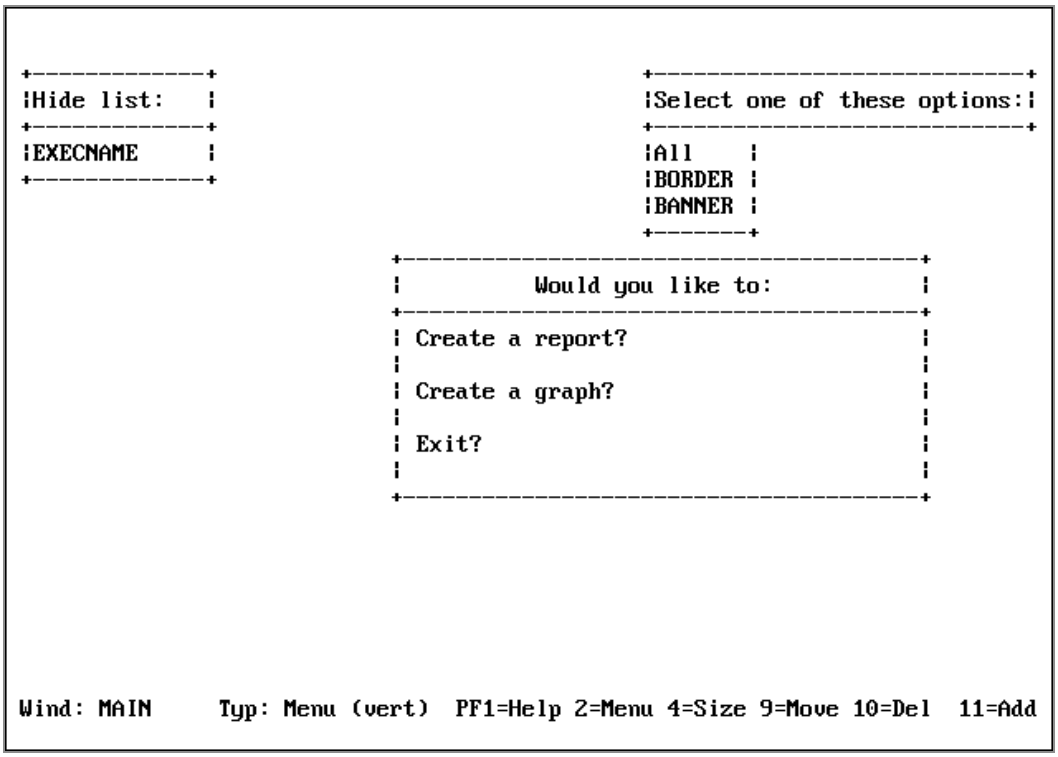
Menu Option	Description
FOCEXEC name	Attaches a FOCEXEC to each menu selection of the window. The FOCEXEC is executed when the menu item is selected.
Heading	Changes the heading of any window you are working on. You can also add or remove a heading.
Description	Changes the description of any window you are working

Menu Option	Description
	on.
Show a window	Used only during window editing, brings another window onto the screen for reference. You cannot edit the second window.
Unshow a window	Removes the shown window from the display.

Menu Option	Description
Display list	Enables you to specify a list of up to 16 windows that are visible when this window is displayed during execution. 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. 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 during execution, windows displayed when executed previously and not hidden afterward, and windows displayed because specified on a display list). The window facility interprets each window heading as a separate window: if all of the windows have headings, 16 can be displayed on the screen at one time.



Menu Option	Description
Hide list	Allows you to specify windows that does 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 are hidden except those in the display list. If you do not hide a window that was displayed, it remains on the screen until another window that includes it on a hide list is displayed during execution.



Menu Option	Description
Popup (Off/On)	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.
Help window	Allows 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 command, you have to explicitly set a PF key as the HELP key, as described in Integrating Windows and the FOCEXEC. When selecting the Help window option, you are asked to supply the name of the Help window file that contains the Help window. Next, you are asked to supply the name of the Help window itself. The Help window can be an existing window, or one that you created.

Menu Option	Description

	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 supplying different help information for each item in a vertical menu, horizontal menu (that is, item-specific help).
	
	To assign item-specific help, use a file contents window that displays a file containing text in the following format
	
	`=>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 that 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.

Help window (*continued*)	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

Menu Option	Description

	of the window and the beginning of the text, 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 assign topic-specific help, but want the help text for some of the topics to be contained in a separate file. In this case, 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.
	
	Use this file-identification format:
	
	`FILENAME= membername ddname`

Help window *(continued)*	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.`

Menu Option	Description

	The DEFAULT text must be the last section in the Help file.
	
	Lines beginning with an asterisk (*) 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.`

Line-break	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.
	
	When you select this option, you see:
	
	`None`
	`New line before value`
	`New line after value`
	`Both`
	
	where:
	
	None
	
	Places return value directly after preceding value. If there is not enough room on this line, return value is placed on the next line.
	
	New line before value

Menu Option	Description
	Places return value on the next line. **New line after value** Places return value on the same line as preceding value. Places next return value on next line. **Both** Places return value on a line by itself.

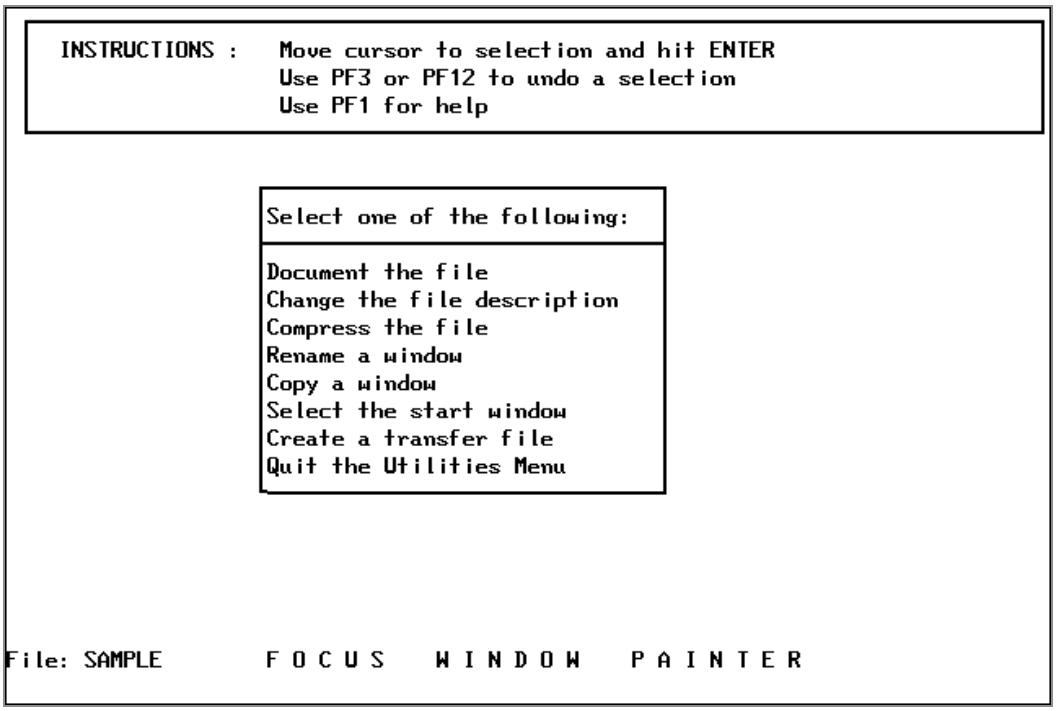
Multi-Select	Enables you to select multiple items from one window. The number of items you select is collected as the return value from that window; each selected item's return value is stored in a temporary file in memory. You can later retrieve these stored values for use in a FOCEXEC. Values for up to 8 windows can be stored at one time.
	When you select this option, you see:
	`-Select Multi(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.

Menu Option	Description
	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.
Multi-Select (*continued*)	The value is assigned to the variable &windowname. 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 need a -WINDOW command with the GETHOLD option. Each value is stored in &windowname. To use this value, assign it to another variable. For example, if the return value has the value 4, issue the special -WINDOW command four times; each time you would collect the value from &windowname. Alternatively, you could perform a loop.

Menu Option	Description
	Note that -WINDOW with the GETHOLD option does not transfer control from the FOCEXEC to the Window facility.
Quit	Returns you to the Window Painter Entry Menu.
Input fields	Input fields pertain to Multi Input Windows. Selecting the field takes you to that field.
Menu text	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.
Text line (x+1)	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.
Pulldown (off/on)	If the setting is ON, placing the cursor on an item in a horizontal menu can display an associated pull-down menu. The default setting is OFF. Turn the setting ON by positioning the cursor on this option and pressing **Enter**. The pull-down menu must be a vertical menu and must be assigned as the goto value for the horizontal menu item. Note that setting Pull-down ON automatically shuts off Menu Text.
Switch window	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.

Utilities Menu

If you select the Utilities option from the Window Painter Main Menu, the Utilities Menu is displayed:



The following table summarizes the options on this menu, along with illustrations of screens that appear when you select some of the options:

Menu Option	Description
Document the file	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. 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 temporarily allocates the PDS. This document contains detailed information about all the windows in the window file. It shows you the kinds of windows, the 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

Menu Option	Description
	individual window description is part of this document. The document is especially useful when creating a FOCEXEC, since it provides return and goto values in addition to other information. **Note:** If you create another file with the same name, the file is not overwritten. It is appended.

```
-* 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,WIDTH=38,WINDOW= 7,POPUP= 0,BORDER= 2,HEADLEN=28,
-* RETURN=None
-* MULTI=Off
-* HEADING:
-*      Would you like to:
-* WINDOW DATA:                GOTOS:          VALUES:
-*  1.'                        ','             ','             '
-*  2.' Create a report?       ','EXECTYPE     ','RPT          '
-*  3.'                        ','             ','             '
-*  4.' Create a graph?        ','EXECTYPE     ','GRPH         '
-*  5.'                        ','             ','             '
-*  6.' Exit?                  ','             ','XXIT         '
-* DISPLAY LIST:
-* BORDER
```

Menu Option	Description
Change the file description	Changes the description of the current window.
Compress the file	This utility is provided to help you save space in memory. It allows space made available by deleted or edited windows to be reused.
Rename a window	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.
Copy a window	This function copies a window from one window file to another, or duplicates it within the same file.

Menu Option	Description
	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 selects 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 another FOCUS environment. 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.

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 FTP.
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 topics.

Creating a Transfer File

The window files that 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.

- 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 use the transfer file during the current FOCUS session: Window Painter temporarily allocates 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 are prompted for the name of the new transfer file. Enter a member name; it can have the same name as the window file, or an entirely new name.

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. See the appropriate operating environment topic in the *Overview and Operating Environments* manual for more information about duplicate window transfer and window documentation file names.

You are asked to select which window(s) you want to transfer. Select

```
All
```

to transfer all of the windows in the current window file, or 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 select the new window. Window Painter merges 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 FTP.

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 the new environment supports features are 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 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. You can accomplish this 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 alphanumeric).

Transfer File Syntax: Window File Attributes

Attribute	Description
FILENAME	The name of the original window file.
DESCRIPTION	A comment field describing the file.
WINDOWNAME	The name of the window.
TYPE	The type of window: 1. Vertical menu 2. Text input window 3. Text display window 4. Horizontal menu 5. File names window 6. Field names window 7. File contents window 8. Return value display window 9. Execution window 10. Multi-input window
COMMENT	A comment field describing the window.
TRANSLATE	Type of input for text input windows (Type 2). 0 Allow mixed-case input. 1 Allow numeric input only. 2 Translate input to uppercase.
ROW	The row number of the upper left corner of the window.
COLUMN	The column number of the upper left corner of the window.

Attribute	Description
HEIGHT	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 what fits in the window frame, use the PF7 and PF8 keys to scroll the window.
TEXT LINE	Position of menu text. Values are: +1, +2, -1, -2.
WIDTH	The width of the window frame, not including the border.
INPUT FIELDS	Fields for multi-input windows.
WINDOW	The number of lines in the actual window frame (not the number of lines of window data). This does not include borders.
POPUP	Sets the pop-up feature. 0 This is not be a pop-up window. 1 This is a pop-up window.

Transfer File Syntax: Window Attributes

Attribute	Description
BORDER	Sets the window border. 0 There is no window border. 1 There is a window border. 2 There is a window border. Options 1 and 2 both result in a basic window border.
HEADLEN	Length of the window heading. If this value is 0, there is no heading.

Attribute	Description
RETURN	Sets the line break feature for use with return value display windows. 0 Line break is not used. 1 New line before this return value. 2 New line after this return value. 3 New line before and after this value.
MULTI	Sets the multi-select feature. 0 This is not a multi-select window. 1 This is a multi-select window.
HEADING	The text of the window heading.
HELP	The name of the help window for this window.
HELPFILE	The name of the window file that contains the help window.
DISPLAY	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.
HIDE	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.

Transfer File Syntax: Window Line Attributes

Attribute	Description
DATA	A line to be displayed in the window (for example, a menu

Attribute	Description
	choice in a vertical menu Window, or a line of text in a text display window). The data can include amper variables (including &windowname).
GOTO	The name of the window to go to if this line is selected by the user. The value can be an amper variable (including &windowname). If the value is blank, and this line is selected, Windows returns to Dialogue Manager.
VALUE	The return value supplied if this line is selected by the user. This value is placed in the amper variable &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 are displayed. For file contents windows (TYPE = 7), this is the name of the file whose contents are to be displayed.

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 occurs. Adjust the ROW and COLUMN attributes if necessary.
- **Screen size.** Windows should not have more than 22 rows or 77 columns. Windows that extend beyond the end of the terminal screen is automatically 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.
- **Blank lines.** Blank line 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 automatically translates references to FOCEXECs, Master Files, and error files, as shown below. You must change other file references yourself when you edit the transfer file.

PC or UNIX Extension	Mainframe ddname
.FEX	FOCEXEC
.MAS	MASTER
.ERR	ERRORS

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.

```
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
COMMENT='Select an existing FOCEXEC from list.'
ROW= 4,COLUMN=11,HEIGHT=11,WIDTH=57,WINDOW=11,POPUP= 0,BORDER=
2,HEADLEN=55,
RETURN=0
MULTI=0
HEADING='Select the request you want to execute and press ENTER:'
DATA='   '
GOTO='        ',VALUE='* FOCEXEC'
$
DISPLAY=BORDER,$
HIDE=BANNER,$
HIDE=MAIN,$
HIDE=EXECTYPE,$
```

Compiling the Transfer File

The transfer file can be executed in its current format, but it may execute slowly, and uses 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 COMPILE 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, 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 temporarily allocates the PDS.

Compile a Transfer File

```
WINDOW COMPILE windowfile
```

where:

windowfile

Is the name of the transfer file.

This must be a member name of a member of a PDS allocated to ddname TRF.

The command creates a new member of the PDS allocated to ddname FMU, with the same member name specified in the command.

When a Dialogue Manager -WINDOW command is encountered in a FOCEXEC, FOCUS searches 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 is used.

Note that if you compile a transfer file and later make changes to it, you need to recompile the updated transfer file: otherwise, FOCUS continues to use the older, unchanged compiled file.

Designing Screens With FIDEL

FIDEL, the FOCUS Interactive Data Entry Language, enables you to design full-screen forms for data entry and application development. You use FIDEL both with MODIFY for building data maintenance and inquiry screens, and with Dialogue Manager for building applications that accept values for variables at run time.

Introduction

Describing the CRT Screen describes the facilities of FIDEL that are common to both MODIFY and Dialogue Manager. This introduction explains how MODIFY facilities and FIDEL interact, and describes the FIDEL facilities that are specific to MODIFY. Using FIDEL in Dialogue Manager describes the interaction between Dialogue Manager and FIDEL.

From the FOCUS TED editor, you can also use the FOCUS Screen Painter with both MODIFY and Dialogue Manager to interactively build and view screens online. With the Screen Painter, you design the layout of the form and the Screen Painter automatically generates the FIDEL code to build it. The FOCUS Screen Painter is described in Using the ibi FOCUS Screen Painter.

The two simple examples on the following pages demonstrate how to generate a screen form by using the CRTFORM and -CRTFORM syntax. Note how closely FIDEL syntax resembles TABLE syntax for creating headings.

Note: FIDEL only supports fixed format records with LRECL=80.

Using FIDEL With MODIFY

The following example of a simple MODIFY CRTFORM illustrates the use of FIDEL with the resulting screen (the numbers refer to the explanation and are *not* part of the code):

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
2.     "EMPLOYEE UPDATE"
```

```
3.     "EMPLOYEE ID #: <EMP_ID   LAST NAME:  <LAST_NAME"
4.     "DEPARTMENT: <DEPARTMENT  SALARY:     <CURR_SAL"
```

```
5.  MATCH EMP_ID
       ON NOMATCH REJECT
       ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
6.  DATA
    END
```

This request sets up a form to update the last name, department and current salary. Processing is as follows:

1. CRTFORM generates the visual form and invokes FIDEL. The form begins on line one of the screen unless specified otherwise with the LINE option (see Using Multiple CRTFORMs: LINE).
2. Each line on the screen begins and ends with double quotation marks. This is a line of text that serves as a title. Note the close correspondence to the syntax used to create headings in a TABLE request.
3. The second screen line specifies two data fields: EMP_ID and LAST_NAME. A data entry field is indicated by a left caret, followed by the field name or alias from the Master File. The text, EMPLOYEE ID #: and LAST NAME: identifies each field on the screen. This informs the operator where to enter the data.
4. This is the last line within double quotation marks. It signals the end of the CRTFORM. In this case it identifies and defines two more data fields: DEPARTMENT and CURR_SAL. When you run the MODIFY request, the form instantly appears on the screen:

   ```
   EMPLOYEE UPDATE
   EMPLOYEE ID #:  LAST NAME:
   DEPARTMENT:  SALARY:
   ```

   The number of characters allotted for each data entry field on the screen defaults to the display format for that particular field in the Master File. You can optionally specify a format for screen display that is shorter than the default.

   The operator can now fill in the data entry areas with the appropriate information.

5. The request continues with MODIFY MATCH logic.
6. This line tells FOCUS that the incoming data is from the terminal. In conjunction with CRTFORM, it implies full-screen data input. You can also use DATA VIA FIDEL.

When you use FIDEL with MODIFY, you are setting up full-screen forms for the maintenance of data source fields. Most MODIFY features, such as conditional and non-conditional fields, automatic application generation, Case Logic, multiple record processing, error handling, validation tests, logging transactions, and typing messages to the terminal, work with FIDEL.

With MODIFY you also have access to additional screen control options such as clearing the screen, specifying and changing the size of the screen, and designating the particular line on which the form starts.

Using FIDEL With Dialogue Manager

The following example of a simple -CRTFORM illustrates the use of FIDEL in Dialogue Manager and the resulting screen (the numbers refer to the explanation and are *not* part of the code):

```
1. -CRTFORM
2. -"MONTHLY SALES REPORT FOR <&CITY/10"
3. -"BEGINNING PRODUCT CODE IS: <&CODE1/3"
   -"ENDING PRODUCT CODE IS: <&CODE2/3"
4. -"REGIONAL SUPERVISOR IS: <&REGIONMGR/5"
   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 PROD_CODE
   IF PROD_CODE IS-FROM &CODE1 TO &CODE2
   FOOTING CENTER
   "REGIONAL SUPERVISOR: &REGIONMGR"
   END
```

The procedure sets up a form for gathering run-time variables for a TABLE request: &CITY, the city for the report; &CODE1 and &CODE2, a range of product codes; and &REGIONMGR, the regional supervisor. Processing is as follows:

1. -CRTFORM generates the visual form, invokes FIDEL, and clears the screen.
2. Each line on the screen begins with a dash and double quotation marks (-"), and ends with double quotation marks. Note this first line of the screen form contains text and a variable field, &CITY, which has a length of 10. This specifies ten spaces on the screen for entering the value. The data entry field is indicated by the left caret.
3. The next few lines of the screen form contain both text and variable fields with formats.
4. The last line within double quotation marks signals the end of the -CRTFORM. When the FOCEXEC executes, the screen displays the following form:

   ```
   MONTHLY SALES REPORT
   FOR
   BEGINNING PRODUCT
   CODE IS:
   ENDING PRODUCT CODE
   IS:
   REGIONAL SUPERVISOR
   IS:
   ```

   The operator can now fill in values for the run-time variables. After the operator transmits the screen by pressing Enter, the values entered on the screen are sent to the variables. The regular FOCUS commands are stacked and executed when the end of the procedure is reached.

When you use FIDEL with Dialogue Manager, you can define input fields as amper variables that receive values at run time to adjust to specific processing requirements. Because they are *not* data fields and are not part of the Master File, they do not automatically have a format. You must allocate space for them on the screen. You can do this directly on the -CRTFORM as in the previous example, or through a -SET statement.

Dialogue Manager supports two additional control statements: -CRTFORM BEGIN and -CRTFORM END. The statement -CRTFORM BEGIN signals the beginning of the screen form. You can then enter screen lines as well as other Dialogue Manager control statements. You then signal the end of the screen form with the statement -CRTFORM END. This allows you to use Dialogue Manager statements between screen lines while building the form.

Screen Management Concepts and Facilities

The following briefly outlines the FIDEL capabilities that are common to both MODIFY and Dialogue Manager and defines the common terminology:

- The MODIFY CRTFORM statement and the Dialogue Manager -CRTFORM control statement both automatically invoke FIDEL. All succeeding lines placed within double quotations make up the actual screen form. Note the common syntax between TABLE headings (see the *Creating Reports* manual) and CRTFORM screen lines.
- You can combine a CRTFORM and a -CRTFORM in one procedure. However, they must remain within their own environments. The MODIFY CRTFORM contains data source fields, whereas the Dialogue Manager -CRTFORM contains amper variables.
- The term *field* in this chapter refers to either a data source field name in conjunction with MODIFY or an amper variable in conjunction with Dialogue Manager.
- You can define a CRTFORM in MODIFY or a -CRTFORM in Dialogue Manager that has more lines than on your CRT screen. FIDEL provides scrolling capabilities.
- It is important to note the difference between the physical screen on the terminal and the logical CRTFORM or form. A form generated by one CRTFORM or -CRTFORM statement can take up many screens or less than one screen.
- You can specify three types of fields on the screen: input, display only, and turnaround (both display and update). Data entry and turnaround fields are considered unprotected areas on the screen because you may input values or replace what is there. Display values are considered protected areas on the screen because you cannot alter what is there (see Data Entry, Display and Turnaround Fields).
- You can set PF key controls and specify cursor positioning. You can specify screen attributes such as background effects, highlighting, and color to enhance readability of the screen. You can also change screen attributes depending on the outcome of various tests (see Describing the CRT Screen, Describing the CRT Screen, and Describing the CRT Screen).

Note: This chapter is written specifically for the IBM 3270 terminal, which supports PF key and cursor control, scrolling and screen attributes.

Using FIDEL Screens: Operating Conventions

The following procedures apply for filling in *all* FIDEL screens:

- To move from field to field, press the Tab key. You can also move the cursor around the screen using the arrow keys.
- When filling in values on the screen, you may use any of the keys on the keyboard. Some terminals automatically prevent the entry of a non-numeric character in a field identified as computational.
- To scroll forward or backward through a long CRTFORM (from screen to screen) press the PF8 or PF7 key, respectively (or PF20, PF19).
- To transmit the screen, press the Enter key.
- If you make an error, the transaction may not be transmitted and an error message may appear at the bottom of the screen. You can correct the error and retransmit the screen.
- To signal the end of data entry, press the PF3 or PF15 key or type END in an unprotected area. In MODIFY, this terminates the request. In Dialogue Manager, this terminates the FOCEXEC procedure.

The following operating procedures are specific to MODIFY:

- To return to the first screen without transmitting the current screen, press the PF2 key or the key set to QUIT.
- If the screen clears at any time, press the Enter key to bring it back.

Note: The PF key settings referred to here are the default settings. Any PF key can be redefined using the SET statement.

Describing the CRT Screen

The MODIFY statement CRTFORM or the Dialogue Manager control statement -CRTFORM, followed by the screen layout, generates a form. Within one MODIFY procedure, you can use an unlimited number of screen lines (within memory constraints). Each screen line can contain a maximum of 78 characters of text and data.

In MODIFY, you can use up to 255 CRTFORM statements in a procedure. In Dialogue Manager, there is no limit to the number of -CRTFORM statements that you may use in one procedure.

All the basic options described here can be used with both MODIFY and Dialogue Manager. Options that are specific to MODIFY are discussed in Using FIDEL in MODIFY and those specific to Dialogue Manager are discussed in Using FIDEL in Dialogue Manager.

The following example shows the syntax of a simple MODIFY CRTFORM using the LOWER case option, followed by two screen lines containing various screen elements: text, a spot marker, and a field (numbers refer to the explanation; they are *not* part of the code):

```
1. CRTFORM LOWER
2. "PLEASE FILL IN THE EMPLOYEE ID # </1"
3. "EMPLOYEE ID #: <EMP_ID"
   MATCH EMP_ID
     .
     .
     .
```

Processing is as follows:

1. CRTFORM invokes FIDEL and generates the form. The LOWER case option specifies that what is entered from the terminal in lowercase will remain in lowercase.
2. The first line of the screen contains descriptive text.

   </1 is a spot marker which skips one blank line.
3. The last line of the screen contains two screen elements: descriptive text that identifies the field and the data source field EMP_ID. The last line between quotation marks signals the end of the CRTFORM.

The form generated appears as follows:

```
PLEASE FILL IN THE EMPLOYEE ID #
```

```
EMPLOYEE ID #:
```

Specifying Elements of the CRTFORM

To create the visual form, you enter the screen lines one after the other within double quotation marks. For each screen line, you can specify various screen elements such as descriptive text and fields. A left caret (<) followed by the name of the field generates the position where data is to be entered onto the screen.

You may need to use two FOCEXEC lines to describe one physical CRTFORM line. Simply omit the double quotation marks (") at the end of the first line and omit them at the

beginning of the next line as well. Everything between the set of double quotation marks will read as one screen line on the CRTFORM.

Invoking FIDEL: CRTFORM and -CRTFORM

The following is a summary of the complete syntax for generating a CRTFORM in MODIFY or a -CRTFORM in Dialogue Manager. The individual options and screen elements are described in detail in specific sections later in the chapter. The syntax is

```
[-]CRTFORM [option option...]
[-]"screen element [screen element....]"
```

where:

[-]CRTFORM

Automatically invokes FIDEL and sets up the visual form. Subsequent lines describe the screen.

option option...

Refers to screen control options. (See Using FIDEL in MODIFY and Using FIDEL in Dialogue Manager.)

[-]"screen element.."

Can be user-defined text, fields, or spot markers. Spot markers define the next place on the screen where a screen element will appear. Both spot markers and fields are preceded by a left caret and optionally closed by a right caret (see Describing the CRT Screen).

Note:

- You can create simple screen forms by typing the FIDEL code into your procedures with your text editor. However, it is easier to build more complex forms using many screen attributes and field labels using the FOCUS Screen Painter.
- You can use the asterisk (*) with CRTFORM in FIDEL to generate a CRTFORM containing all of the data source's fields automatically (that is, without specifying individual fields). See Using FIDEL in MODIFY for information on CRTFORM *, its syntax and variations.

- Do not begin any field used in a CRTFORM or FIXFORM statement with X*n*, where *n* is any numeric value. This applies to fields in the Master File and computed fields.

Defining a Field

Labels, prefixes, attributes, and formats are parts of the definition of a particular field. In Dialogue Manager, the first character is an ampersand, which signals an amper variable. (The entire definition is preceded by a left caret and optionally closed by a right caret.)

Note: Fields with a text (TX) format cannot be used in CRTFORM or -CRTFORM. However, they can be entered interactively using TED (see Entering Text Data Using TED, for using text fields in MODIFY).

Define a Field in FIDEL

The syntax for defining a field is as follows.

In MODIFY:

```
<[:label.][prefix.][attribute.]field[/length][>]
```

In Dialogue Manager:

```
<[&:label.][prefix.][attribute.]&variable[/length][>]
```

where:

:label.|&:label.

Is a user-defined label of up to 12 characters associated with a field. It may not contain embedded blanks (see Describing the CRT Screen).

prefix.

Refers to D. or T., which designate a display or turnaround field, respectively (see Data Entry, Display and Turnaround Fields).

attribute.

Is the abbreviation or full name of a screen attribute (see Describing the CRT Screen).

field

Is the name of the field or variable being defined.

&variable

Is for data entry. Can be a data source field or a temporary field.

/length

Is the length of the field as it appears on the screen. In MODIFY, you need to define a length only if you want the screen length to be different from the format length that is defined in the MASTER or COMPUTE. In Dialogue Manager, you need to define a length only if not previously defined.

Note: When you use the abbreviations for attributes, you do not need to use the dot separator between attributes or between a prefix and an attribute (see Describing the CRT Screen).

Defining a Field

The following is an example of the syntax of a Dialogue Manager screen line defining the variable field &CITY:

```
-CRTFORM
-"<&:L01.T.HIGH.&CITY/7"
        .
        .
        .
```

The elements on the second line which define the variable field &CITY are:

1. The left caret generates a place for the variable on the screen.
2. &:L01 is a label that identifies the data entry area on the screen (see Describing the CRT Screen).
3. T. is a prefix that defines the variable as a turnaround field. If the variable has been given a value within the FOCEXEC, it is displayed. Otherwise a default value is displayed. The operator can then change the value.
4. .HIGH. is a screen attribute specifying that the contents of the field will be highlighted.

5. &CITY/7 is the name of the variable field with a length specification. The specified length is seven characters. That is, the space that will be allotted on the screen for input of data is seven characters long.

Prefixes, labels, and screen attributes are explained fully in Data Entry, Display and Turnaround Fields, Describing the CRT Screen, and Describing the CRT Screen.

Difference in FIDEL When Used With MODIFY and Dialogue Manager

The following chart outlines the similarities and differences of FIDEL when used with MODIFY and Dialogue Manager:

MODIFY	Dialogue Manager
`CRTFORM [options]`	`-CRTFORM [options]`

`UPPER/LOWER`
`CLEAR/NOCLEAR`
`WIDTH/HEIGHT`
`TYPE`
`LINE`
	`UPPER/LOWER`
	`BEGIN/END`
	`TYPE`

`"screen elements"`
`text`
 	
`<spot marker[>]**`
`<field/length[>]*`
`prefix.(D. or T.)***`
`attribute.`
`:label.`
	`"screen elements"`
	`text`
	
	`<spot marker[>]**`
	`<field/length[>]**`
	`prefix.(D. or T.)***`
	`attribute`
	`&:label.`

* The right caret denotes a non-conditional field.

** The right caret has no meaning, but may be used for increased clarity.

*** Prefixes, attributes and labels are part of the definition of the field on the screen. They do not stand alone.

Using Spot Markers for Text and Field Positioning

Because the lengths of fields vary, text does not automatically align uniformly on the screen. Spot markers are available to help you position both text and fields. Please note that a spot marker is essential to eliminate trailing blanks at the end of the first line, if your screen line description takes up two FOCEXEC lines.

The syntax and usage of the different spot markers are shown in the following chart:

Marker	Example	Usage
`<n or <n>`	`<50`	Positions the next character in column 50.
`<+n or <+n>`	`<+4`	Positions the next character four columns from the last non-blank character.
`<-n or <-n>`	`<-1`	Positions the next character one column to the left of the last character. This marker's function is to suppress or write over the attribute byte at the beginning and the end of a field.
`</n or </ n>`	`</2`	Positions the next character at the beginning of the line that is two lines from the last (skips two lines). **Note:** The last line is blank and is created when a double quotation mark (") is encountered.
`<0X or <0X>`	`<0X`	Positions the next character immediately to the right of the last character (skip zero columns). This is used to help position data on a FIDEL screen when a single screen line is coded as two lines in a FOCEXEC. No

Marker	Example	Usage
		spaces are inserted between the spot marker and the start of a continuation line (see Note 3 in the following example).

Note: You can optionally use the right caret >. This is useful when the next character in the line is a left caret. It also enhances readability.

Suppose you want the various input data fields arranged across the screen in vertical sections, left justified, and in horizontal segments marked off with lines. Using spot markers, you can create the desired screen as shown in the following example:

```
   MODIFY FILE EMPLOYEE
   CRTFORM
    "EMPLOYEE UPDATE"
1.  "</1"
    "-------------------------------------------------------------"
    "EMPLOYEE ID #: <EMP_ID   LAST NAME: <LAST_NAME"
1.  "</1"
2.  "DEPARTMENT: <DEPARTMENT <+3 CURRENT SALARY:<0X>
    <CURR_SAL"
    "-------------------------------------------------------------"
    "BANK: <BANK_NAME"
    "-------------------------------------------------------------"
   MATCH EMP_ID
   .
   .
   .
   DATA
   END
```

The spot markers in the example perform the following functions:

1. </1 generates a blank line.
2. <+3 moves the word CURRENT three spaces to the right of the last letter in the word DEPARTMENT. <0X> skips no spaces. No extra spaces are inserted between this and the next word (<CURR_SAL) on the continuation line. There is, in fact, one space before the field which is an attribute byte that marks the start of a field.

The screen appears as:

```
EMPLOYEE UPDATE


----------------------------------------
EMPLOYEE ID #:  LAST NAME:


DEPARTMENT:  CURRENT SALARY:
----------------------------------------
BANK:
----------------------------------------
```

Specifying Lowercase Entry: UPPER/LOWER

All text that is entered from the terminal is normally translated to uppercase letters. You can override this default and preserve both uppercase and lowercase text by using the lowercase option. The syntax is

```
[-]CRTFORM [UPPER|LOWER]
```

where:

UPPER

Translates all characters to uppercase. This is the default.

LOWER

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

Note: In MODIFY, when you use multiple CRTFORMs on the same screen (using LINE n), you can mix UPPER and LOWER among the forms.

Data Entry, Display and Turnaround Fields

There are three types of data or variable fields that can be specified on the CRTFORM: data entry, display, and turnaround.

You can also compute data fields (see Computing Values: The COMPUTE Statement, for rules about computing data fields) and specify them as entry, display, or turnaround on the CRTFORM. You can convert a turnaround field to a display field dynamically.

In MODIFY, fields can also be designated as conditional or unconditional (see Conditional and Non-Conditional Fields). We recommend that for data entry, you use conditional fields (left caret only) so that the values in your data source are not replaced by a blank or a zero if you do not enter data for the field.

For most turnaround fields, we recommend that you use non-conditional fields (both carets). A non-conditional turnaround field remains active whether you enter data or not. Because the value in the data source is displayed in the field, that value remains in the data source if you do not change it. Because the field remains active, the values for your VALIDATEs and COMPUTEs are then accurate (see Conditional and Non-Conditional Fields for a complete explanation of the use of conditional and non-conditional fields in MODIFY).

The following outlines the rules for specification of different types of fields.

Use Data Entry Fields (for Data Entry Only)

In MODIFY, the syntax is

```
<field[/length][>]
```

where:

<field[>]

Is the name of the field. Reserves space on the screen for data entry into that field and does not display the current value of the field.

In MODIFY, if only the left caret is used, data entry is conditional. If both carets are used, the field is non-conditional (see Conditional and Non-Conditional Fields).

In Dialogue Manager the syntax is

```
<&variable[/length][>]
```

where:

<&variable[>]

Is the name of the variable field. Reserves space on the screen for data entry into that field and does not display the current value of the field.

In Dialogue Manager, the option of the right caret is meaningless. Usually for the FOCEXEC to run, you must supply a value for each variable. If you do not, FOCUS assumes a blank or a 0 for that value.

Use Display Fields (for Information Only)

Data is displayed in a protected area and cannot be altered.

In MODIFY, the syntax is

```
<D.field[/length]
```

In Dialogue Manager, the syntax is

```
<D.&variable[/length]
```

where:

D.

Is the prefix placed in front of a field, indicating that the data or value is to be displayed. The current value of the field appears on the screen, but in a protected area which cannot be changed.

Note that the right caret is meaningless for display fields.

Use Turnaround Fields (for Display and Change)

Data is displayed in an unprotected area and can be altered.

In MODIFY, the syntax is:

```
<T.field[/length][>]
```

In Dialogue Manager, the syntax is:

```
<T.&variable[/length][>]
```

where:

T.

Is the prefix placed in front of a field to indicate that it is a turnaround field. The current value of the field is displayed on the screen. However, the operator may change the value, as it is not in a protected area.

In MODIFY, if only the left caret is present, the T. field is treated as conditional. If the right caret is used, the field is non-conditional, and the value is treated as present, even if unchanged (see Conditional and Non-Conditional Fields).

In Dialogue Manager, the changed value for the turnaround variable field will substitute everywhere in the FOCEXEC where it is subsequently encountered.

Note: In MODIFY, in order to display data from a data source field or present it for turnaround, a position in the data source must first be established through the use of a MATCH or NEXT statement, or value must be assigned in a COMPUTE. A computed field cannot be set and displayed in the TOP case, where data entry is processed prior to computations. For example, one of the phrases

```
ON MATCH CRTFORM
ON NEXT CRTFORM
```

must be used. A position is thus established in the data source, and the values of the fields in existing records are now available for display as protected or unprotected fields.

You can also match on a key field and go to a case (see Using FIDEL in MODIFY) in which you display a CRTFORM using display and turnaround fields.

Using Data Entry, Display, and Turnaround Fields

This section will show how to use Date Entry, Display, and Turnaround Fields with MODIFY and Dialogue Manager.

Using Data Entry, Display, and Turnaround Fields With MODIFY

The following example combines two CRTFORMs in a single MODIFY request and shows the use of entry, display and turnaround fields (numbers refer to the explanation below; they are *not* part of the code):

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
       "ENTER EMPLOYEE ID#: <EMP_ID"
       "PRESS ENTER"
       "</2"
2.  MATCH EMP_ID
       ON NOMATCH REJECT
       ON MATCH CRTFORM
          " "
          "REVISE DATA FOR SALARY AND DEPARTMENT"
          "ENTER NEW DATA FOR EDUCATION HOURS"
          " "
3.        "EMPLOYEE ID #: <D.EMP_ID   LAST_NAME: <D.LAST_NAME"
          " "
4.        "SALARY:    <T.CURR_SAL>"
          "DEPARTMENT: <T.DEPARTMENT>"
5.        "EDUCATION HOURS: <ED_HRS>"
       ON MATCH UPDATE CURR_SAL DEPARTMENT ED_HRS
    DATA
    END
```

The procedure matches the employee ID, displays both the ID and the last name, and then displays the current salary and department for turnaround. Education hours is a data entry field.

Note that when the procedure executes, both CRTFORMs are displayed immediately. However, the display and turnaround fields in the second CRTFORM do not display data until the operator fills in the first form and presses Enter. We therefore recommend you use the LINE option.

When a FORMAT ERROR occurs, all data entered up to that point is processed and cannot be changed in the course of your transaction.

The processing is as follows:

1. CRTFORM generates the first form which begins on line 1 (the second CRTFORM is displayed, but without values):

```
ENTER EMPLOYEE ID #:
PRESS ENTER

REVISE DATA FOR SALARY AND DEPARTMENT
ENTER NEW DATA FOR EDUCATION HOURS

EMPLOYEE ID #:   LAST NAME:
SALARY:
DEPARTMENT:
EDUCATION HOURS:
```

2. The procedure continues with the MATCH logic. If the ID number that is input matches an ID in the data source, the display and turnaround fields on the second CRTFORM display the data. Assume the operator enters 818692173 and presses Enter.

   The following is displayed:

```
ENTER EMPLOYEE ID #: 818692173
PRESS ENTER

REVISE DATA FOR SALARY AND DEPARTMENT
ENTER NEW DATA FOR EDUCATION HOURS

EMPLOYEE ID #: 818692173     LAST NAME: CROSS
SALARY:   27062.00
DEPARTMENT:  MIS
EDUCATION HOURS:
```

3. This screen line contains two display fields.
4. The next two screen lines contain turnaround fields.
5. The last line is a data entry field.

Note: To display fields from a unique segment, the ON MATCH CONTINUE TO, ON NEXT, or MATCH WITH-UNIQUES phrase must have been executed (see Modifying Data: MATCH and NEXT).

In Dialogue Manager, in order to display values with D. or T., a value must have been supplied for the variable prior to the initiation of the -CRTFORM. System variables are an exception to this rule, as the system automatically supplies their values.

Computed fields in both MODIFY and Dialogue Manager can be displayed in any kind of CRTFORM.

Using Data Entry, Display, and Turnaround Fields With Dialogue Manager

The following example illustrates the use of D. fields and system variables in a Dialogue Manager -CRTFORM:

```
-SET &CITY = STAMFORD;

-CRTFORM
-"YEARLY SALES REPORT FOR <T.&CITY/10"
-"DATE: <D.&DATE  TIME: <D.&DATEMDYY"
_" "
-"ENTER BEGINNING PRODUCT CODE RANGE: <&BEGCODE/3"
-"ENTER ENDING PRODUCT CODE RANGE: <&ENDCODE/3"
-"ENTER NAME OF REGIONAL SUPERVISOR: <&REGIONMGR/15"
```

```
TABLE FILE SALES
```

```
HEADING CENTER
"YEARLY REPORT FOR &CITY"
"PRODUCT CODES FROM &BEGCODE TO &ENDCODE"
" "
SUM UNIT_SOLD AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
BY PROD_CODE
IF PROD_CODE IS-FROM &BEGCODE TO &ENDCODE
IF CITY EQ &CITY
FOOTING CENTER
"REGION MANAGER: &REGIONMGR"
"CALCULATED AS OF &DATE"
END
```

The example processes as follows:

1. The -SET sets a default value for &CITY:

```
 FOR WHICH CITY DO YOU WANT A REPORT?
```

2. -CRTFORM generates the screen form:

```
YEARLY SALES REPORT FOR STAMFORD
DATE: 02/22/2003    TIME: 13.42.38

ENTER BEGINNING PRODUCT CODE RANGE:
ENTER ENDING PRODUCT CODE RANGE:
ENTER NAME OF REGIONAL SUPERVISOR:
```

3. The transaction value for &CITY is Stamford, the value that was previously supplied in the -SET statement.
4. Note that the variables &DATE and &DATEMDYY are system variables. The values are supplied by the system and displayed on the form.

Controlling the Use of PF Keys

The terminal operator can use certain PF keys to control the execution of a FIDEL application. Normally, the following keys are used:

- PF3 and PF15 mean END and terminate execution.
- PF2 means Cancel and cancels the transaction in MODIFY.
- PF7 and PF8 page Backward and Forward respectively.

Note: All other keys return the value of the PF key when pressed.

Several facilities are available to assist you in controlling various screen operations:

- You can reset PF key functions. You can also set PF keys to branch to particular cases in MODIFY or labels in Dialogue Manager.
- You can set the cursor on a specified position on the screen (see Describing the CRT Screen).

- You can use the cursor position on the screen to perform a branch or action based on a test (see Describing the CRT Screen).

Default Settings for PF Keys

The default PF key settings are as follows:

PF Key	Function
PF01	HX
PF02	CANCEL
PF03, PF15	END
PF04, PF16	RETURN
PF05, PF17	RETURN
PF06, PF18	RETURN
PF07, PF19	BACKWARD
PF08, PF20	FORWARD
PF09, PF21	RETURN
PF10, PF22	RETURN
PF11, PF23	RETURN
PF13	RETURN
PF12, PF24	UNDO
PF14	RETURN

You can display the current PF key settings by issuing the FOCUS query command:

```
? PFKEY
```

This displays a formatted table of all the current values.

Resetting PF Key Controls

You can reset PF key functions in FIDEL for both CRTFORMs and -CRTFORMs using the FOCUS SET command with the following syntax

```
SET PFxx = function
```

where:

xx

Is a one or two-digit PF key number.

function

Is one of the following:

END in MODIFY, exits the procedure; in Dialogue Manager, is equivalent to QUIT. That is, END exits the procedure.

CANCEL in MODIFY, cancels the transaction and returns to the TOP case. Do not use the CANCEL setting in Dialogue Manager.

FORWARD pages forward.

BACKWARD pages backward.

RETURN has no specific screen action. Returns the PF key name in the PFKEY field because it is not yet defined. To set the PFKEY field, use COMPUTE in MODIFY or -SET in Dialogue Manager.

HELP displays text supplied with the HELPMESSAGE attribute for any field on the MODIFY CRTFORM. Position the cursor on the data entry area of the desired field, and press the PF key you have defined for HELP. If no help message exists for that field, the following message is displayed:

```
NO HELP AVAILABLE FOR THIS FIELD.
```

The following example sets the PF03 key for paging backward and the PF04 key for paging forward:

```
SET PF03=BACKWARD,PF04=FORWARD
```

Note: When changing PF key settings, make sure that at least one key is set to END. If you set a PF key to FORWARD, you should also set one to BACKWARD.

Setting PF Key Fields for Branching Purposes

You can create a menu of processing options. The operator can then indicate a choice by pressing a particular PF key. To assign a specific processing function to a PF key, you must specify a field named PFKEY. Which PF key the operator presses determines the value of the PFKEY field.

You can use the PF keys designated as Return keys, as well as the Enter key. You define a variable called PFKEY (in MODIFY) or &PFKEY (in Dialogue Manager) and then test its value after the CRTFORM is displayed. Which branch takes place depends on which PFKEY the operator presses.

In MODIFY, the syntax is

```
COMPUTE
PFKEY/A4=;
```

where:

PFKEY/A4

Is a four-character field, whose value is determined by which key the operator presses at run time.

In Dialogue Manager, the syntax is

```
-SET &PFKEY='  ';
```

where:

&PFKEY

Is a four-character field, whose value is determined by which key the operator presses at run time.

=' ';

Is the allocation of four character spaces for the field.

The following example shows how PF keys can be tested in MODIFY:

```
1.  COMPUTE
    PFKEY/A4=;
2.  CRTFORM
       "SELECT OPTION"
       "INPUT   PRESS PF4"
       "UPDATE  PRESS PF5"
       "DELETE   PRESS PF6"
3.  IF PFKEY EQ 'PF04' GOTO INCASE
       ELSE IF PFKEY EQ 'PF05' GOTO UPCASE
       ELSE IF PFKEY EQ 'PF06' GOTO DELCASE
       ELSE GOTO TOP;
            .
            .
            .
```

The example processes as follows:

1. The COMPUTE statement specifies a four-character field PFKEY.
2. CRTFORM generates the form which supplies the operator with three options:

   ```
   SELECT OPTION
   INPUT   PRESS PF4
   UPDATE  PRESS PF5
   DELETE  PRESS PF6
   ```

3. The IF test determines what case to branch to depending on the value of the PFKEY field. For example, if the operator presses PF4, the value for PFKEY is PF04, and the request branches to an input case INCASE.

Specifying Screen Attributes

Screen attributes (such as highlighting, colors, and so on) can be applied to the fields on the CRTFORM and the -CRTFORM. They can also be used as background effects and can be applied to the fields depending on the result of tests.

The following attributes are available on 3270 IBM terminals:

Function	Abbreviation	Short Name
Flash or Blink	F	FLAS or BLIN
Underline	U	UNDE
Invert or Reverse Video	I	INVE or REVV
Clear*	C	CLEA
Blue	B	BLUE
Red	R	RED
Pink	P	PINK
Green	G	GREE
Aqua	A	AQUA
Turquoise	T	TURQ
Yellow	Y	YELL
White	W	WHIT
Nodisplay*	N	NODI
Return to default	$	$
Highlight or Intensify*	H	HIGH or INTE

Note:

- *Clear, Nodisplay, and Highlight or Intensify can be used on all terminals. Clear also sets the highlight off for entry and turnaround fields. Nodisplay is not supported for D. or T. fields. The remaining attributes are also known in the FOCUS community as extended attributes.
- Use of abbreviations is recommended, except for TURQ.

When an attribute is unsupported on a particular terminal or is specific to a version of FOCUS under another operating system, the attribute is ignored. Therefore, there is no need for code changes between terminals and/or operating systems.

To use the screen attributes other than C, N, and H you must notify FOCUS that your terminal is equipped to display them. Issue the FOCUS SET command:

```
SET EXTTERM=ON
```

This allows a procedure to be operated on a variety of terminals. FOCUS automatically detects a 3279 model terminal and sets EXTTERM to ON by default.

If your terminal does not properly recognize extended attributes, due to a "terminfo" compatibility problem, stray characters will appear on your screen. You may turn off extended attribute recognition with the command:

```
SET EXTTERM=OFF
```

Programs with extended attributes and EXTTERM=OFF will run as if extended attributes had not been coded in the program.

Make sure that your terminal has the extended attribute options needed before you turn EXTTERM on. There are many different IBM 3270 models. Generally, the color terminals in the 3279 series have most of the options. However, even if a terminal has the physical capability to support all of the attributes, it may be defined to the operating system as a lower grade terminal. In such cases, you must ascertain whether or not all the attributes can be used.

The syntax for defining screen attributes in MODIFY is

```
<[:label][.attribute.]field[>]
```

The syntax for defining screen attributes in Dialogue Manager is

```
<[&:label][.attribute.]&variable[>]
```

where:

.attribute.

Is one or more of the attributes. Note the dots (periods) before and after each attribute or entry in an attribute list.

field

Names the field to which the attributes apply.

&variable

Names the variable field to which the attributes apply.

Note: Labels and their use are discussed in Describing the CRT Screen.

The following chart shows you how to use these attributes in conjunction with prefixes (D. and T.), where X is the name of a field or variable:

`.HT.X`	Highlighted T.
`.CT.&X`	Unhighlighted T.
`.N.X`	Nodisplay entry, (for example, for passwords)
`.H.&X`	Highlighted entry
`.C.X`	Unhighlighted entry
`.HD.X`	Highlighted D.

The following usage considerations apply when using screen attributes:

- An attribute stays in effect until another attribute changes it.
- A list of attributes may be composed entirely of abbreviations in any order. If abbreviations only are used, you do not need the dot separator between attributes.
- The last mentioned option in a group of mutually exclusive attributes will be taken.
- A color or flash overrides a highlight, clear, or Nodisplay.
- If short names are used, the first four letters identify the attribute. Each name must be separated by a dot. Either abbreviations or short names can be used, but they cannot be mixed without a dot separator.
- Full names may be used as well. Each must be delimited by a dot.
- You can change screen attributes during the course of a terminal session by using labeled fields.

Note the following examples:

`.AID.`	Aqua inverted display field.
`<.RED.FLASH.`	Red flashing field.
`<.RED.FLAS.`	Red flashing field.
`<.PIN.`	Inverted pink field (color overrides).
`<I.YELL.`	Inverted yellow field.

Using Background Effects

If a field is absent, the attribute affects the protected portion of the screen; that is, the text. Both a beginning and ending dot as well as a space between the attribute and the text are needed. For example:

```
"<.RED. ENTER EMP_ID:"
```

This will print the words ENTER EMP_ID: in red. Note the space between .RED. and ENTER EMP_ID:. A right caret may also be inserted for clarity.

The line:

```
"<.INVE.RED.   <.CLEAR.EMP_ID"
```

will turn the background color to red. CLEAR changes the background for the input field EMP_ID back to black.

An attribute stays in effect until another attribute changes it on a physical screen. Therefore, if <.INVE.RED. is in the upper left corner, the entire screen will be in inverse red unless some other background attribute is provided later. In the example above, the <.CLEAR is used to limit the effect to one area.

Note: .CLEAR. and .HIGH. only work when they are used in conjunction with a field. They do not work alone or simply with text.

Using Labeled Fields

You can use labels to identify a specific field on the screen. They are necessary to perform the following functions:

- Dynamically change screen attributes during processing depending on the results of tests.
- Position the cursor on the screen, or read the position of the cursor on the screen, where there is no pre-existing field.

The syntax for a labeled field in MODIFY is

```
<:label.field
```

The syntax for a labeled field in Dialogue Manager is

```
<&:label.&variable
```

where:

<[&]:label.

Is a user-defined label. It starts with a colon (:) and may be up to 66 characters long including the colon. You may not use embedded blanks.

field

Is any field on the CRTFORM. It can be a field created specifically for appending a label.

&variable

Is any variable field on the CRTFORM. It can be a field created specifically for appending a label.

The following rules apply:

- A label cannot occur by itself. It must be used with a field.
- A label must be declared using a COMPUTE, -SET, or -DEFAULTS statement.
- Setting a label to $ returns its field to the default attribute.

Using a Labeled Field With MODIFY

For example, in MODIFY:

```
COMPUTE
:ONE/A6='   ';
CRTFORM
"<:ONE.EMP_ID"
```

The label :ONE is set to a format of A6 and is the identifier of the field EMP_ID.

Using a Labeled Field With Dialogue Manager

For example, in Dialogue Manager:

```
-SET &:ONE='  ';
-CRTFORM
-"<&:ONE.&CITY/10"
```

In this Dialogue Manager example, the label &:ONE is set to a format of A4 and is the identifier of the field &CITY.

Note: When you are dealing with many complex labels and attributes, we advise you to use the FOCUS Screen Painter which allows you to do everything without learning the detailed syntax (see Using the ibi FOCUS Screen Painter).

Dynamically Changing Screen Attributes

The screen attributes in a FIDEL form can be changed during the course of the terminal session in which they are defined. This allows you to design easy-to-read and easy-to-use procedures. For instance, after an error occurs, you can turn a specific field into flashing red to alert the operator.

The mechanism for changing the attribute is to put a label before the field. Then, issue a COMPUTE in MODIFY, or a -SET in Dialogue Manager, to assign the label new attribute values. When the screen is next displayed, it takes on the characteristics of the provided attributes.

The following example shows how to use a COMPUTE in MODIFY to dynamically change an attribute value:

```
COMPUTE
 :ATTRIB/A12=IF CURR_SAL GT 50000 THEN 'FLASH' ELSE '$';
CRTFORM
 "AMOUNT <:ATTRIB.T.CURR_SAL>"
IF CURR_SAL GT 50000 GOTO TOP ELSE GOTO OTHER;
 .
 .
 .
```

This generates an attribute value for the label ATTRIB. If the CURR_SAL is greater than 50,000, the field will flash; otherwise, it observes the default setting.

The following example shows the use of a -SET statement to assign an attribute value in Dialogue Manager:

```
-SET &AMOUNT=0;
-SET &:ATTRIB='     ';
-TOP
-CRTFORM
-"AMOUNT: <&:ATTRIB.T.&AMOUNT>"
-SET &:ATTRIB=IF &AMOUNT GT 100 THEN 'FLASH' ELSE '$';
-IF &AMOUNT GT 100 GOTO TOP;
```

```
.
.
.
```

This generates an attribute value for the label &:ATTRIB, changing &AMOUNT to flashing if the value is greater than 100. Be sure to use -SET to establish the label in the beginning of the procedure.

Note: When you use CRTFORMs in either MODIFY or Dialogue Manager, the labels you assign must precede the fields with which they are associated; labels cannot occur by themselves. Use COMPUTE statements to dynamically change screen text attributes, setting the label equal to the COMPUTE (see previous example).

You can convert a T. field to a D. field dynamically; however, you cannot convert a D. field to a T. field. The method for changing turnaround fields to display fields is the same as that for changing screen attributes dynamically.

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
2.  "SALARY UPDATE"
2.  " "
3.  "EMPLOYEE ID #: <.INVE.EMP_ID LAST NAME: <0X
    <.CLEAR.D.LAST_NAME"
4.  MATCH EMP_ID
       ON NOMATCH REJECT
5.     ON MATCH CRTFORM LINE 10
6.     ENTER SALARY"
       " "
       "SALARY: <:HERE.T.CURR_SAL>"
7.  COMPUTE
       :HERE/A12=IF CURR_SAL GT 100000 THEN 'D' ELSE 'T';
       IF CURR_SAL GT 100000 GOTO TOP;
       ON MATCH UPDATE CURR_SAL
    DATA
    END
```

This procedure constructs a form to update salaries. It processes as follows:

1. CRTFORM generates the screen form and invokes FIDEL.
2. Provide text for the CRTFORM; empty quotation marks indicate a blank line on the form.
3. The next two lines contain the following screen elements:

EMPLOYEE ID #:

Is text describing the conditional data field EMP_ID.

.INVE.

Is a screen attribute that displays the field EMP_ID in reverse video.

LAST NAME:

Is text describing the field LAST_NAME.

.CLEAR.

Is a screen attribute that clears the .INVE. attribute, returning the D. (display-only) field LAST_NAME to the default display.

4. The request continues with MODIFY MATCH logic.
5. If EMP_ID matches, another CRTFORM is generated on line 10 of the same screen.
6. The next three lines contain the following screen elements:

   ENTER SALARY:

   Is text describing the CURR_SAL field.

   " "

   Generates a blank line.

   :HERE

   Is a label identifying the CURR_SAL field.

7. This COMPUTE evaluates the field CURR_SAL and defines it as a turnaround (T.) field or a display (D.) field, depending on the value of CURR_SAL. If the salary is greater than 100,000, the field is a display field (and cannot be updated); if the salary is less than 100,000, the field is a turnaround field (and can be updated).

The resulting CRTFORM is as follows:

```
SALARY UPDATE

EMPLOYEE ID #:     LAST NAME:
```

```
ENTER SALARY

SALARY:
```

Specifying Cursor Position

To specify cursor position, simply choose the field where you want the cursor positioned. You may specify the field by its field name or by its label. You can set the cursor at a specific place on the screen by computing or setting the value of the field CURSOR (in MODIFY) or &CURSOR (in Dialogue Manager).

The syntax for the field which controls the cursor position in MODIFY is

```
COMPUTE
CURSOR/A66= expression;
```

where:

CURSOR/A66

Is a 66-character alphanumeric field.

expression

Is terminated with a semicolon and can be anything, including the full field name, its full alias, or a unique truncation of either, or the label itself. This determines the position of the cursor.

For example:

```
COMPUTE
CURSOR/A66=IF TESTNAME GT 100 THEN 'EMP_ID'
ELSE 'LAST_NAME';
```

The position of the cursor will be on the field EMP_ID if the value of test name is greater than 100, or it will be on the field LAST_NAME if test name is less than or equal to 100.

You may also position the cursor using a field label. For example:

```
COMPUTE
CURSOR/A66=IF TESTNAME GT 100 THEN ':ONE'
ELSE ':TWO';
```

Note: If the field name is not unique, FIDEL uses the first occurrence of the field name (going from left to right across each line and then down to the next line) to set or test the cursor position.

In MODIFY, the variable CURSORINDEX can also be used to compute the position of the cursor at a particular record when there are multiple indexed records displayed in a single CRTFORM. This feature is commonly used for placing the cursor on invalid fields after VALIDATE statements. The syntax is

```
COMPUTE
CURSORINDEX/I5=expression;
```

where:

CURSORINDEX/I5

Is a five-digit integer field. Refers to the current value of the subscript being processed from the CRTFORM.

expression

May be any expression, but in most applications will be set equal to REPEATCOUNT.

Note: See Case Logic, Groups, CURSORINDEX and VALIDATE for a full example of the use of CURSORINDEX using Case Logic, multiple fields and the VALIDATE subcommand. Also, multiple record processing is discussed in full in Multiple Record Processing.

In Dialogue Manager, the syntax for positioning the cursor is

```
-SET &CURSOR=expression;
```

where:

&CURSOR

Is a variable specifically referring to the position of the cursor.

expression

Is terminated with a semicolon and can be any valid expression including the field name or label itself. It determines the position of the cursor.

The following example illustrates the positioning of the cursor on the screen in Dialogue Manager using labeled fields:

```
1.  -SET &:AAA = '    ';
    -SET &:BBB = '    ';
2.  -PROMPT &YR.PLEASE ENTER YEAR NEEDED.
3.  -SET &CURSOR = IF &YR GT 1984 THEN ':AAA' ELSE ':BBB';
    -*
4.  -CRTFORM
    -"MONTHLY REPORT FOR THE CITY <&:AAA.&CITY/10"
    -"YEARLY REPORT FOR THE AREA <&:BBB.&AREA/1"
        .
        .
        .
```

This processes as follows:

1. Two -SET statements declare the labels, which are themselves variables.
2. The -PROMPT statement prompts the operator for a value for &YR.
3. The -SET statement sets an IF test as the value for the variable &CURSOR. If the value of &YR is greater than 1984, the position of the cursor is set to the label :AAA; otherwise, it is set to the label :BBB.
4. If the operator supplies the value 85 for &YR, the visual form generated is as follows, and the cursor is positioned at the variable &CITY:

```
MONTHLY REPORT FOR THE CITY
YEARLY REPORT FOR THE AREA
```

The remainder of the FOCEXEC might then branch to a TABLE request for a monthly report for that city. Had the year been earlier than 84, the cursor would have been positioned on AREA. The branch might then be to a TABLE request for a yearly report for that area.

Caution: In Dialogue Manager, be sure to set &CURSOR to the label name without the & (ampersand). Use :AAA, not &:AAA.

Determining Current Cursor Position for Branching Purposes

Rather than having the operator type a response, you can create a menu on which you list options. To select an option, the operator moves the cursor to the correct line on the screen and presses the Enter key. FOCUS senses the cursor position and takes action based upon it (such as branching to a particular case or field).

To do this, you must specify a 66 character field that contains the current cursor position, CURSORAT. You may identify a field on the screen by a label or by its field name.

The syntax that defines the field used to read the cursor position in MODIFY is

```
COMPUTE
CURSORAT/A66=;
```

where:

CURSORAT/A66

Is the field whose value is determined by the field name, or label of the field, on which the cursor is positioned when the operator presses Enter.

In Dialogue Manager, the syntax is

```
-SET &CURSORAT='      ';
```

where:

&CURSORAT

Is a variable whose value is determined by the field name, or label of the field, on which the cursor is positioned when the operator presses Enter.

If the actual cursor position is not on any field, the value of CURSORAT is the nearest preceding field. If there are no preceding fields, the value of CURSORAT is the TOP of the CRTFORM. That is, the value is at the very beginning of the CRTFORM.

In the following example, field XYZ is a computed field for the purpose of creating a labeled field wherever necessary on the CRTFORM:

```
     MODIFY FILE EMPLOYEE
1.   COMPUTE
     CURSORAT/A66=;
2.   :ADD/A1=;
     :UPP/A1=;
3.   XYZ/A1=;
4.   CRTFORM
        "POSITION CURSOR NEXT TO OPTION DESIRED"
        "THEN PRESS ENTER"
        " "
        "<:ADD.XYZ   ADD RECORDS"
        "<:UPP.XYZ   UPDATE RECORDS"
5.   IF CURSORAT EQ ':ADD' GOTO ADD ELSE
     IF CURSORAT EQ ':UPP' GOTO UPP ELSE GOTO TOP;

      CASE ADD
     CRTFORM LINE 1
        "THIS CRTFORM ADDS RECORDS"
        " "
        "EMPLOYEE ID #: <EMP_ID"
        "LAST NAME:     <LAST_NAME"
        "FIRST NAME:    <FIRST_NAME"
        "HIRE DATE:     <HIRE_DATE"
        "DEPARTMENT:    <DEPARTMENT"
     MATCH EMP_ID
        ON MATCH REJECT
        ON NOMATCH INCLUDE
     ENDCASE

     CASE UPP
     CRTFORM LINE 1
        "THIS CRTFORM UPDATES RECORDS"
        " "
        "EMPLOYEE ID #: <EMP_ID"
        "DEPARTMENT:    <DEPARTMENT"
        "JOB CODE:      <CURR_JOBCODE"
        "SALARY:        <CURR_SAL"
     MATCH EMP_ID
        ON NOMATCH REJECT
        ON MATCH UPDATE DEPARTMENT CURR_JOBCODE CURR_SAL
     ENDCASE
     DATA
     END
```

This example processes as follows:

1. The COMPUTE establishes the field CURSORAT.
2. The second and third COMPUTEs declare the labels :ADD and :UPP.
3. The third COMPUTE establishes a field XYZ for the purpose of using labels.
4. CRTFORM generates the following visual form beginning on the first line of the screen:

```
POSITION CURSOR NEXT TO OPTION DESIRED
THEN PRESS ENTER

ADD RECORDS
UPDATE RECORDS
```

5. An IF phrase tests the value of the field CURSORAT. If the operator places the cursor next to ADD RECORDS, the value of CURSORAT is :ADD and a branch to Case ADD will be performed. If the operator places the cursor next to UPDATE RECORDS, the value of CURSORAT is :UPP and Case UPP will be performed.

Annotated Example: MODIFY

The following example illustrates the syntax for a MODIFY CRTFORM using dynamically changing attributes:

```
MODIFY FILE EMPLOYEE
1. CRTFORM
2. "EMPLOYEE UPDATE"
3. "</1"
4. "EMPLOYEE ID #: <.INVE.EMP_ID"
GOTO UPDATE
CASE UPDATE
5. MATCH EMP_ID
ON NOMATCH REJECT
6. ON MATCH CRTFORM LINE 1
" "
7. "LAST NAME: <.INVE.T.LAST_NAME"
"DEPARTMENT: <.CLEAR.T.DEPARTMENT>"
"SALARY: <:ATTRIB.T.CURR_SAL>"
8. ON MATCH COMPUTE
:ATTRIB/A12 = IF CURR_SAL GT 50000 THEN 'FLASH.INVE';
```

```
MSG/A60 = IF CURR_SAL GT 50000 THEN 'PLEASE REENTER' ELSE ' ';
ON MATCH TYPE "<MSG"
ON MATCH IF CURR_SAL GT 50000 GOTO UPDATE;
ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
ENDCASE
DATA
END
```

This procedure sets up a form to update the department and current salary. It processes as follows:

1. CRTFORM generates the visual form and invokes FIDEL.
2. This line contains a screen element set between double quotations marks. It is a line of descriptive text.
3. This line contains another screen element, a spot marker that skips one line.
4. These lines contain four screen elements—'EMPLOYEE ID #:' is text describing the field; the field EMP_ID is described as a conditional data entry field. The contents will be displayed in reverse video because .INVE. is a screen attribute defining the field.

   The visual form generated is as follows:

   ```
   EMPLOYEE UPDATE
   EMPLOYEE ID #: (reverse video)
   ```

   Enter Employee ID # 818692173.
5. The request continues with MODIFY MATCH logic.
6. If the EMP_ID matches, another CRTFORM is generated. It is placed on LINE 1 and thus replaces the previous CRTFORM on the screen.
7. The CRTFORM defines three turnaround fields:

   The LAST_NAME field. The .INVE. attribute displays the field in reverse video.

   The DEPARTMENT field. The .CLEAR. attribute displays the field in regular video.

   The CURR_SAL field. The appearance of the field value depends on the value of the :ATTRIB field. When the CURR_SAL value first appears, the :ATTRIB field is empty and the value appears in regular video. If you enter a CURR_SAL value greater than 50,000, the :ATTRIB field receives the attribute FLASH.INVE, displaying the CURR_SAL

value in flashing inverse (or reverse) video. The CRTFORM appears as follows:

```
LAST NAME:   CROSS
DEPARTMENT:  MIS
SALARY:      27062.00
```

8. If the CURR_SAL field value is greater than 50,000 when you press Enter, the COMPUTE statement assigns the :ATTRIB label the FLASH.INVE attribute.
9. If the CURR_SAL field value is greater than 50,000 when you press Enter, the IF statement branches back to the CASE UPDATE statement. This displays the second CRTFORM with the CURR_SAL value in reverse video.

   Note: If you are using a terminal emulator you may not be able to view the attribute FLASH.INVE.

Annotated Example: Dialogue Manager

The following sample -CRTFORM illustrates the syntax for dynamic control of attributes in Dialogue Manager:

```
1.  -PROMPT &CITY.FOR WHICH CITY DO YOU WANT A REPORT?.
2.  -SET &:ATTRIB = IF &CITY EQ STAMFORD THEN 'INVE' ELSE 'CLEAR';
    -*
3.  -CRTFORM
4.  -"MONTHLY SALES REPORT"
5.  -"Date: <D.&DATE      Time: <D.&TOD"
6.  -"Beginning Code is:  <&:ATTRIB.&BEGCODE/3"
    -"Ending Code is:   <&:ATTRIB.&ENDCODE/3"
    -"Regional Supervisor is:  <&:ATTRIB.&REGIONMGR/15"
    TABLE FILE SALES
    HEADING CENTER
    "MONTHLY REPORT FOR &CITY"
    "PRODUCT CODES FROM &BEGCODE TO &ENDCODE"
    " "
    SUM UNIT_SOLD AND RETURNS AND COMPUTE
    RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
    BY PROD_CODE
    IF PROD_CODE IS-FROM &BEGCODE TO &ENDCODE
    IF CITY EQ &CITY
```

```
    FOOTING CENTER
    "REGION MANAGER: &REGIONMGR"
    "CALCULATED AS OF &DATE"
7.  END
```

The example processes as follows:

1. The -PROMPT prompts the operator for a value for &CITY.
2. The -SET statement sets the label :ATTRIB to INVE if the city is Stamford, causing each field labeled :ATTRIB in the remainder of the -CRTFORM to be displayed in reverse video.
3. -CRTFORM generates the visual form and invokes FIDEL.
4. The first line of the screen form contains text.
5. The second line contains the current date and time as display fields. Since they are in protected areas of the screen, they cannot be altered.
6. Each of the next three lines contains descriptive text and one field. Each field has a label which displays the field in reverse video if the city is Stamford.

   The screen displays the following -CRTFORM:

   ```
   MONTHLY SALES REPORT
   Date: 01/08/97    Time: 10:50:16
   Beginning Code is:
   Ending Code is:
   Regional Supervisor is:
   ```

7. After the operator presses Enter, the values entered in the screen form are sent to the variables. The TABLE request executes when END is encountered.

Using FIDEL in MODIFY

The following standard MODIFY functions and concepts work with FIDEL in the building of CRTFORMs (for additional information on these functions):

- Conditional and non-conditional field specification (see Conditional and Non-

Conditional Fields).

- The FIXFORM statement which can be used before the first CRTFORM. This enables you to mix data sources (see Using FIDEL in MODIFY).
- Automatic application generation, which enables you to use several simple statements to generate automatic data management procedures and CRTFORMs (see Using FIDEL in MODIFY).
- Multiple CRTFORMs for different processing options. The additional FIDEL facility of the LINE option helps you organize the use of multiple CRTFORMs (see Using Multiple CRTFORMs: LINE).
- Case Logic, which enables you to divide the processing into logical subdivisions for particular sets of circumstances (see Case Logic, and Using FIDEL in MODIFY).
- Groups of fields (see Using FIDEL in MODIFY).
- VALIDATES and various error handling formats (see Using FIDEL in MODIFY).
- Log files that preserve a record of all data that is entered onto the screen (see Using FIDEL in MODIFY).

MODIFY also has additional screen control options such as clearing the screen, setting the height and width parameters, and changing the default size of the TYPE message area in order to enlarge the CRTFORM (see Using FIDEL in MODIFY).

Conditional and Non-Conditional Fields

When you run a MODIFY request, FOCUS keeps track of which transaction fields are active or inactive during execution. In order to add, update, and delete segment instances, the fields must be active (see Active and Inactive Fields, for a full discussion of active and inactive fields).

You can define data entry and turnaround fields as either conditional or non-conditional. A conditional field is conditionally active. That is, it becomes active only if there is incoming data present for the field. Otherwise, it remains inactive. A non-conditional field is always active whether there is incoming data present or not.

When you are performing update operations, there are several important points to keep in mind when you choose whether to specify a field as conditional or non-conditional:

- If data is entered or changed, the data source value is always updated and the field

always becomes active. This is true whether the field is conditional or non-conditional.

- If data is not entered or changed, what happens to the data source value is dependent on whether the field is conditional or non-conditional as well as program logic. The following table outlines this.

Type of Field	Active/Inactive	Data Source Value
Conditional Entry	Inactive	Remains. Display value ignored.
Conditional Turnaround	Inactive	Remains. Display value ignored.
Non-Conditional Entry	Active	Displayed value replaces data source value (blank or 0).
Non-Conditional Turnaround	Active	Displayed value replaces data source value (same value).

- If a field is active, the displayed value always becomes the new data source value. In turnaround fields, this is by definition the same value.
- If a field is inactive, the displayed value is always ignored.
- If you compute a data source field and then display it on the CRTFORM with a D. or a T., the field must still be active to get the computed value displayed on the screen. Otherwise, you get a blank or 0.
- When you use a VALIDATE for a field, the field must be active. Otherwise you do not get the accurate data source value validated; instead, you get a blank or 0.

  Note: You can make a field active or inactive by using the ACTIVATE or DEACTIVATE statement respectively.

Conditional and Non-Conditional Display and Turnaround Fields

The following example illustrates the display and turnaround field features as well as the use of a non-conditional turnaround field (both carets). The first CRTFORM asks for a key

field value, in this case EMP_ID. If a matching record is obtained, then some data source values are displayed and others are shown for turnaround update.

Note how the non-conditional turnaround field functions in the following example. Whether the displayed value is changed or not, the value in the data source is active. The VALIDATE uses the display value, whether it was changed or not.

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
       "ENTER EMPLOYEE ID#: <EMP_ID"
       "PRESS ENTER BEFORE CONTINUING"
       "-------------------------------------------------"
    MATCH EMP_ID
       ON NOMATCH TYPE
          "EMPLOYEE ID NOT IN DATABASE. PLEASE REENTER."
       ON NOMATCH REJECT
2.     ON MATCH CRTFORM LINE 4
          " "
          "EMPLOYEE ID #:  <D.EMP_ID"
          "LAST NAME:  <D.LAST_NAME"
          "HIRE DATE:  <D.HIRE_DATE"
          "SALARY:  <T.CURR_SAL>"
          "DEPARTMENT:  <T.DEPARTMENT>"
3.     ON MATCH VALIDATE
          SALTEST = IF CURR_SAL GT 0 THEN 1 ELSE 0;
          ON INVALID TYPE
          "SALARY MUST BE GREATER THAN 0"
          "CORRECT SALARY AND PRESS ENTER TWICE"
       ON MATCH UPDATE CURR_SAL DEPARTMENT
    DATA
    END
```

The example processes as follows:

1. When the procedure executes, the top part of the CRTFORM appears as follows:

   ```
   ENTER EMPLOYEE ID #:
   PRESS ENTER BEFORE CONTINUING
   -----------------------------------------------------
   ```

   If the employee ID entered does not match an ID in the data source, the transaction

is rejected and a TYPE statement appears at the bottom of the screen.

Assume the operator enters the following employee ID:

```
ENTER EMPLOYEE ID #: 818692173
PRESS ENTER BEFORE CONTINUING
------------------------------------------------------
```

2. If the ID entered matches an ID in the data source, FOCUS successfully retrieves a record. The ON MATCH CRTFORM causes a second CRTFORM to be displayed on line 4. This CRTFORM contains both display and turnaround fields. The data source values of the fields appear on the second CRTFORM, and the cursor is positioned at the start of the CURR_SAL field which is the first unprotected field. Note that both CURR_SAL and DEPARTMENT are automatically highlighted for turnaround:

```
ENTER EMPLOYEE ID #: 818692173
PRESS ENTER BEFORE CONTINUING
------------------------------------------------------

EMPLOYEE ID #:   818692173
LAST NAME:       CROSS
HIRE DATE:       811102
SALARY:          27062.00
DEPARTMENT:      MIS
```

Assume the operator updates DEPARTMENT, does not change CURR_SAL, and transmits the CRTFORM:

```
ENTER EMPLOYEE ID #: 818692173
PRESS ENTER BEFORE CONTINUING
------------------------------------------------------

EMPLOYEE ID #:   818692173
```

```
LAST NAME:       CROSS
HIRE DATE        811102
SALARY:          27062.00
DEPARTMENT:      ois
```

3. When the operator presses Enter, the transaction is processed. If the value of CURR_SAL is greater than 0, the VALIDATE will evaluate as 1 (true) and processing continues. Although a value was not entered for CURR_SAL, the field is active because it is specified as a non-conditional field. Thus, the VALIDATE reads the current value in the T. field (27062.00), and validates the field. The transaction is then processed.

If you specify the turnaround field as conditional (only the left caret), the field is inactive if no data is entered. Assume the same transaction as above. The operator updates the DEPARTMENT and does not enter new data for the CURR_SAL field. The VALIDATE does not read the T. value because the field is inactive and instead reads a 0. The field is invalidated and the following error message occurs:

```
ENTER EMPLOYEE ID #: 818692173
PRESS ENTER BEFORE CONTINUING
--------------------------------
```

```
EMPLOYEE ID #:   818692173
LAST NAME:       CROSS
HIRE DATE:       811102
SALARY:          27062.00
DEPARTMENT:      ois
```

```
(FOC421)TRANS 1 REJECTED INVALID SALTEST
INVALID SALARY
SALARY MUST BE GREATER THAN 0
```

Using FIXFORM and FIDEL in a Single MODIFY

A MODIFY procedure can mix data sources from CRTFORMs and FIXFORMs.

The rules are:

- You can have only one FIXFORM statement and you must specify the name of the transaction data source. For example:

```
FIXFORM ON filename
```

- The FIXFORM statement must precede the CRTFORM statement.
- START and STOP do not apply (see Reading Selected Portions of Transaction Data Sources: The START and STOP Statements).

This feature is useful in situations where a known set of records is wanted and the keys for these records reside on an external fixed format data source. (The data source may have been produced by a prior TABLE and SAVE or HOLD command.) The procedure first reads a key, fetches the matching record, and displays it on the CRTFORM specified.

This is also convenient when the FIXFORM is included in a START case.

In the following example, FIXFORM is used with FIDEL. To run this example on your machine, you must first create a sequential data source with data. To do so, run this TABLE request:

```
TABLE FILE EMPLOYEE
PRINT EMP_ID PAY_DATE
IF PAY_DATE GE 820730
ON TABLE SAVE AS PAYTRANS
END
```

This creates the transaction data source PAYTRANS. Then run the following MODIFY request:

```
    MODIFY FILE EMPLOYEE
1.  FIXFORM ON PAYTRANS EMP_ID/9 PAY_DATE/6
2.  MATCH EMP_ID
      ON NOMATCH REJECT
      ON MATCH CONTINUE
    MATCH PAY_DATE
3.    ON MATCH/NOMATCH CRTFORM
         "EMPLOYEE ID #:<D.EMP_ID"
         "PAY DATE:<D.PAY_DATE"
         "MONTHLY GROSS:<T.GROSS>"
      ON NOMATCH INCLUDE
      ON MATCH UPDATE GROSS
```

```
DATA
END
```

The example processes as follows:

1. First the data is read in from the sequential data source PAYTRANS.
2. The EMP_ID from PAYTRANS is matched against EMP_IDs in the EMPLOYEE data source. If the EMP_IDs match, PAY_DATE is matched.
3. The CRTFORM shows display values for EMP_ID and PAY_DATE. If there is a match on PAY_DATE, GROSS is displayed as a turnaround field and the operator can update it. If there is no match on PAY_DATE, both PAY_DATE and GROSS are included:

```
EMPLOYEE ID #:   071382660
PAY_DATE:        820831
MONTHLY GROSS:         916.67
```

The procedure ends when there are no more transactions to read on the external data source. It can also be terminated by the operator by pressing the PF1 or PF3 key.

Generating Automatic CRTFORMs

You can use several simple but powerful statements with the FOCUS MODIFY facility to allow immediate generation of data management requests. You do not need to learn the complete FOCUS MODIFY language. Without using field names, you can write general-purpose requests and customize them for more detailed situations.

The statements can be used with multi-segment data sources as well as simple data sources. They can also be used from the Screen Painter (see Using the ibi FOCUS Screen Painter). These statements automatically specify conditional fields. They include:

`CRTFORM * [SEG n]`	Design screen for all real data fields in segment *n*, where *n* is either the segment name or number.

`CRTFORM * KEYS [SEG n]`	Design screen for all key fields in segment *n*.
`CRTFORM * NONKEYS [SEG n]`	Design screen for all non-key fields in segment *n*.
`CRTFORM T.* [SEG n]`	Design screen using T.fields in segment *n*
`CRTFORM D.* [SEG n]`	Design screen using D.fields in segment *n*.

Note: The use of CRTFORM * on a COMBINE data source name is illogical and may produce unpredictable results.

Note that you can optionally specify the segment name or number for each of the CRTFORMs. To obtain the segment names and numbers, enter the following command where *file* is the name of the data source:

```
CHECK FILE file PICTURE
```

The names and numbers appear on the top of each segment in the diagram. You may also list segment names and numbers by entering the command:

```
? FDT filename
```

See the *Describing Data* manual and the *Developing Applications* manual for more information on the CHECK FILE command and ? FDT query.

If you are modifying all of the segments in the data source (except for unique segments), you can write the request without using Case Logic. The following example adds and maintains data for the EMPLOYEE data source. The segments are as follows:

- Segment 1 contains basic employee data (names, jobs, salaries, and so on).
- Segment 3 contains employee salary histories.
- Segment 7 stores employees' home addresses and information on their bank accounts.

- Segment 8 stores employee monthly pay.
- Segment 9 stores monthly deductions.

(Segment 2 is a unique segment. Segments 4, 5, and 6 are cross-referenced segments that are not part of the EMPLOYEE data source.)

The request is:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "THIS PROCEDURE ADDS NEW RECORDS AND UPDATES EXISTING RECORDS </1"
   "INSTRUCTIONS"
   "1. ENTER DATA FOR EACH FIELD"
   "2. USE TAB KEY TO MOVE CURSOR"
   "3. PRESS ENTER WHEN FINISHED"
   "4. WHEN YOU FINISH ALL RECORDS, PRESS PF1 </1"
CRTFORM * KEYS
MATCH * KEYS SEG 01
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 01
   ON MATCH UPDATE * SEG 01
   ON NOMATCH INCLUDE
MATCH * KEYS SEG 03
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 03
   ON MATCH UPDATE * SEG 03
   ON NOMATCH INCLUDE
MATCH * KEYS SEG 07
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 07
   ON MATCH UPDATE * SEG 07
   ON NOMATCH INCLUDE
MATCH * KEYS SEG 08
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 08
   ON MATCH UPDATE * SEG 08
   ON NOMATCH INCLUDE
MATCH * KEYS SEG 09
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 09
   ON MATCH UPDATE * SEG 09
   ON NOMATCH INCLUDE
DATA
END
```

When the procedure executes, the screen appears as follows:

```
 THIS PROCEDURE ADDS NEW RECORDS AND UPDATES EXISTING RECORDS
```

```
INSTRUCTIONS
1. ENTER DATA FOR EACH FIELD
2. USE TAB KEY TO MOVE CURSOR
3. PRESS ENTER WHEN FINISHED
4. WHEN YOU FINISH ALL RECORDS, PRESS PF1

EMP_ID:      :
DAT_INC:     :
TYPE:    :
PAY_DATE:     :
DED_CODE:    :

LAST_NAME:        :                    FIRST_NAME:        :
HIRE_DATE:     :                       DEPARTMENT:        :
CURR_SAL:       :                      CURR_JOBCODE:    :
ED_HRS:     :

PCT_INC:     :                         SALARY:        :
JOBCODE:   :

ADDRESS_LN1:         :
ADDRESS_LN2:         :
ADDRESS_LN3:         :

ACCTNUMBER:      :

GROSS:       :
```

Notice that the fields are divided into five groups. The first group consists of all the key fields in the data source. Each subsequent group consists of all non-key fields in a particular segment. Fill in each group from top to bottom and press Enter before filling in the next group. When you do this, the request uses the values to match on the segments specified later in the request.

The first CRTFORM statement generates the first group of fields, which are all the key fields in the data source:

```
CRTFORM * KEYS
```

The MATCH statements in the request modify each of the segments in the data source. Each statement contains a CRTFORM phrase that prompts for all non-key fields in the segment:

```
CRTFORM T.* NONKEYS SEG xx
```

Note that the CRTFORM phrase displays the fields as turnaround fields. After you fill in the fields in the group and press Enter, FOCUS uses the field values to update the segment.

You can add the following enhancements to the request:

- The LINE option on each CRTFORM statement.
- Explanatory text after each CRTFORM statement.
- A separate CRTFORM containing explanatory text at the beginning of the request.

If you want to modify some but not all segments in the data source, use Case Logic to write the request. Place each MATCH statement in a separate case. For example, this request modifies data in Segments 1, 3, and 7:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "THIS PROCEDURE MAINTAINS EMPLOYEE"
   "JOB DATA, SALARY HISTORIES, AND ADDRESSES"
   " "
CRTFORM * KEYS
   "FILL IN EMP_ID, DAT_INC, AND TYPE FIELDS"
   "THEN PRESS ENTER"
GOTO EMPLOYEE

CASE EMPLOYEE
MATCH * KEYS SEG 01
   ON NOMATCH REJECT
   ON MATCH CRTFORM T.* NONKEYS SEG 01 LINE 10
   ON MATCH UPDATE * SEG 01
   ON MATCH GOTO MONTHPAY
ENDCASE

CASE MONTHPAY
MATCH * KEYS SEG 03
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 03 LINE 10
   ON MATCH UPDATE * SEG 03
```

```
   ON MATCH GOTO DEDUCT
   ON NOMATCH INCLUDE
   ON NOMATCH GOTO DEDUCT
ENDCASE

CASE DEDUCT
MATCH * KEYS SEG 07
   ON MATCH/NOMATCH CRTFORM T.* NONKEYS SEG 07 LINE 10
   ON MATCH UPDATE * SEG 07
   ON NOMATCH INCLUDE
ENDCASE
DATA
END
```

Using Multiple CRTFORMs: LINE

You can choose which screen line the CRTFORM will begin on by using the LINE option. By default, the first CRTFORM begins on line 1. The next CRTFORM in the procedure begins on the line following the end of the previous CRTFORM. For example, if one screen uses 12 lines, the next CRTFORM automatically begins on the 13th line.

In the following example, there are two logical forms: EMPLOYEE UPDATE and FUND TRANSFER INFORMATION UPDATE. It illustrates the placement of CRTFORMs when the default is in effect (that is, the LINE option is not used):

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
       "EMPLOYEE UPDATE"
       " "
       "-------------------------------------------------"
       "EMPLOYEE ID #: <EMP_ID    LAST_NAME: <LAST_NAME"
       "
       "DEPARTMENT:    <DEPARTMENT <28 SALARY: <CURR_SAL"
       " "
       "BANK: <BANK_NAME"
       " "
       "FILL IN THE ABOVE FORM AND PRESS ENTER"
       "-------------------------------------------------"
    MATCH EMP_ID
       ON NOMATCH REJECT
       ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
       ON MATCH CONTINUE TO BANK_NAME
```

```
         ON NOMATCH INCLUDE
2.       ON MATCH/NOMATCH CRTFORM
       "</1"
       "FUND TRANSFER INFORMATION UPDATE"
       " "
       "-------------------------------------------------"
       "BANK: <D.BN   ACCT #: <T.BA"
       " "
       "BANK CODE: <T.BC <30 START DATE: <T.EDATE"
       "-------------------------------------------------"
           ON MATCH UPDATE BA BC EDATE
DATA
END
```

This produces the following screen when the request is executed:

```
EMPLOYEE UPDATE

------------------------------------------------------
EMPLOYEE ID #:           LAST_NAME:

DEPARTMENT:              SALARY:

BANK:

FILL IN THE ABOVE FORM AND PRESS ENTER
------------------------------------------------------

FUND TRANSFER INFORMATION UPDATE

------------------------------------------------------
BANK:                    ACCT #:

BANK CODE:               START DATE:
------------------------------------------------------
```

Note that when the default is in effect, each logical form is displayed one after the other on the screen, the instant the MODIFY procedure is executed. That is, all the distinct CRTFORMs are concatenated into one visual form.

The LINE option enables you to control both the placement of a CRTFORM on the screen and the timing with which it appears on the screen. Using LINE gives you the following options:

- You can have one logical form replace another after each transaction by having subsequent CRTFORMs begin on the same line.
- You can build mixed screens by saving lines from a previous CRTFORM on the screen while executing a subsequent CRTFORM. In other words, the first CRTFORM is displayed, the operator transmits the data, and the next CRTFORM is displayed while the previous one remains on the screen.

The syntax is

```
CRTFORM [LINE nn]
```

where:

nn

Is the starting line number for the CRTFORM.

To completely replace one screen with the next, both CRTFORMs must start on the same line. Note the following change in the previous example:

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
       "EMPLOYEE UPDATE"
       " "
       "-------------------------------------------------"
       "EMPLOYEE ID #:  <EMP_ID LAST_NAME: <LAST_NAME"
       " "
       "DEPARTMENT:     <DEPARTMENT <30 SALARY: <CURR_SAL"
       " "
       "BANK:           <BANK_NAME"
       " "
       "FILL IN THE ABOVE FORM AND PRESS ENTER"
       "-------------------------------------------------"
    MATCH EMP_ID
      ON NOMATCH REJECT
      ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
```

```
      ON MATCH CONTINUE TO BANK_NAME
      ON NOMATCH INCLUDE

2.    ON MATCH/NOMATCH CRTFORM LINE 1
      "</1"
      "FUND TRANSFER INFORMATION UPDATE"
      " "
      "-------------------------------------------------"
      "BANK: <D.BN ACCT #: <T.BA"
      " "
      "BANK CODE: <T.BC <30 START DATE: <T.EDATE"
      "-------------------------------------------------"
      ON MATCH UPDATE BA BC EDATE
    DATA
    END
```

1. When the MODIFY procedure is executed, the following screen is displayed:

```
EMPLOYEE UPDATE

-------------------------------------------------------
EMPLOYEE ID #:          LAST_NAME:

DEPARTMENT:             SALARY:

BANK:

FILL IN THE ABOVE FORM AND PRESS ENTER
-------------------------------------------------------
```

2. After the operator enters and transmits the data, the next CRTFORM replaces the previous one on the screen:

```
FUND TRANSFER INFORMATION UPDATE

-------------------------------------------------------
BANK:                    ACCT #:

BANK CODE:               START DATE:
-------------------------------------------------------
```

Generally, it is a good practice to put LINE 1 on all CRTFORMs that start a new case (see Using FIDEL in MODIFY) unless a specific screen pattern is wanted.

A combination of two or more individual CRTFORMs can occupy specific lines on one screen. To obtain a mixed screen, place the desired starting line number with the CRTFORM statement. For instance:

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
      "EMPLOYEE UPDATE"
      " "
      "-------------------------------------------------"
      "EMPLOYEE ID #:    <EMP_ID LAST_NAME: <LAST_NAME"
      " "
      "DEPARTMENT:       <DEPARTMENT <30 SALARY: <CURR_SAL"
      " "
      "BANK:             <BANK_NAME"
      " "
      "FILL IN THE ABOVE FORM AND PRESS ENTER"

      "-------------------------------------------------"
```

```
    MATCH EMP_ID
      ON NOMATCH REJECT
      ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
      ON MATCH CONTINUE TO BANK_NAME
      ON NOMATCH INCLUDE
2.    ON MATCH/NOMATCH CRTFORM LINE 12
      "</1"
      "FUND TRANSFER INFORMATION UPDATE"
      " "
      "-------------------------------------------------"
      "BANK: <D.BN  ACCT #: <T.BA"
      " "
      "BANK CODE: <T.BC <30 START DATE: <T.EDATE"
      "-------------------------------------------------"
      ON MATCH UPDATE BA BC EDATE
DATA
END
```

Processing occurs as follows:

1. Like the preceding examples, this produces the first screen. Assume the operator enters and transmits the following data:

   ```
   EMPLOYEE UPDATE
   ```

```
-----------------------------------------------------
EMPLOYEE ID #:   117593129          LAST_NAME:   JONES

DEPARTMENT:   MIS                   SALARY:   18480

BANK:   STATE

FILL IN THE ABOVE FORM AND PRESS ENTER
-----------------------------------------------------
```

2. The first CRTFORM remains on the screen while the next CRTFORM is displayed on line 12:

```
EMPLOYEE UPDATE

-----------------------------------------------------------
EMPLOYEE ID #:   117593129            LAST_NAME:   JONES

DEPARTMENT:   MIS                     CURRENT SALARY:   18480

BANK:   STATE

FILL IN THE ABOVE FORM AND PRESS ENTER
-----------------------------------------------------------

FUND TRANSFER INFORMATION UPDATE
-----------------------------------------------------------
BANK: STATE                           ACCT #:40950036

BANK CODE: 510271                     START DATE:821101
```

```
-------------------------------------------------------
```

You can save certain lines from the preceding CRTFORM while you discard others. In the previous example, if you begin the second CRTFORM on line 6, the EMP_ID and the LAST_NAME remain and the next line is the beginning of the FUND TRANSFER INFORMATION AND UPDATE.

Assume the operator enters and transmits data on the first CRTFORM. Part of the first logical form disappears and the second form is displayed. Thus, a new visual form is created:

```
EMPLOYEE UPDATE

-------------------------------------------------
EMPLOYEE ID #:   117593129        LAST_NAME:   JONES

FUND TRANSFER INFORMATION AND UPDATE

-------------------------------------------------
BANK:  STATE                      ACCT #:   40950036

BANK CODE:  510271                START DATE:   821101
-------------------------------------------------
```

You can create mixed screens using the LINE option, in a variety of ways, depending on the need of the application.

CRTFORMs and Case Logic

Case Logic, described in Case Logic, enables you to perform separate complete MODIFY processes in one procedure. Each of these is a case, and the procedure contains directions about which case to execute under various circumstances.

When you use the Case Logic option of the MODIFY command, you can create a pattern of many CRTFORMs.

When there are multiple CRTFORMs in a single MODIFY request, use the LINE option to specify where each CRTFORM will be displayed. With Case Logic, generally, we recommend that you use LINE 1 to replace one screen with another.

The following example illustrates the use of Case Logic with the CRTFORM:

```
MODIFY FILE EMPLOYEE
COMPUTE
   PFKEY/A4= ;
CRTFORM
   "TO INPUT A NEW RECORD, PRESS PF4"
   "TO UPDATE AN EXISTING RECORD, PRESS PF5"
IF PFKEY EQ 'PF04' GOTO ADD ELSE
IF PFKEY EQ 'PF05' GOTO UPP ELSE GOTO TOP;

CASE ADD
CRTFORM LINE 1
   "EMPLOYEE ID #:    <EMP_ID"
   "LAST NAME:   <LAST_NAME FIRST NAME: <FIRST_NAME"
   "HIRE DATE:   <HIRE_DATE"
   "DEPARTMENT:       <DEPARTMENT"
MATCH EMP_ID
   ON MATCH REJECT
   ON NOMATCH INCLUDE
ENDCASE

CASE UPP
CRTFORM LINE 1
   "EMPLOYEE ID #:    <EMP_ID"
   "DEPARTMENT:       <DEPARTMENT"
   "JOB CODE:    <CURR_JOBCODE"
   "SALARY:      <CURR_SAL"
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH UPDATE DEPARTMENT CURR_JOBCODE CURR_SAL
ENDCASE
DATA
END
```

The first CRTFORM appears as:

```
TO INPUT A NEW RECORD, PRESS PF4
TO UPDATE AN EXISTING RECORD, PRESS PF5
```

If the operator presses PF4, the following is displayed:

```
EMPLOYEE ID #:
LAST NAME:              FIRST NAME:
HIRE DATE:
DEPARTMENT:
```

If the operator presses PF5, the following is displayed:

```
EMPLOYEE ID #:
DEPARTMENT:
JOB CODE:
SALARY:
```

Note: At the end of a MODIFY procedure, control defaults to the TOP Case.

Specifying Groups of Fields

Groups of fields (that is, more than one occurrence of the same field) can be specified on the CRTFORM in two ways:

- Specifying a field more than once on a CRTFORM.
- Using REPEAT syntax.

You can use Case Logic to process groups of fields.

Specifying Groups of Fields for Input

A group of fields may repeat on the form. For example:

```
"EMPLOYEE ID   DEPARTMENT   SALARY"
"<EMP_ID       <DPT         <CURR_SAL"
"<EMP_ID       <DPT         <CURR_SAL"
"<EMP_ID       <DPT         <CURR_SAL"
```

This reads the same data as the FIXFORM statement:

```
FIXFORM 3(EMP_ID/C9 DPT/C10 CURR_SAL/C14)
```

The following example shows the use of repeating groups for a single employee ID:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID #: <EMP_ID"
   " "
   "ENTER PAY DATE AND GROSS PAY FOR ABOVE EMPLOYEE"
   " "
   "PAY DATE: <PAY_DATE      GROSS: <GROSS"
   "PAY DATE: <PAY_DATE      GROSS: <GROSS"
   "PAY DATE: <PAY_DATE      GROSS: <GROSS"
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH CONTINUE
MATCH PAY_DATE
   ON MATCH REJECT
   ON NOMATCH INCLUDE
DATA
END
```

Note: A group of repeated data fields cannot be specified on a MATCH or NOMATCH CRTFORM. They must be presented on a primary CRTFORM (that is, one not generated as a result of a MATCH or NOMATCH command).

This procedure processes as follows:

```
ENTER EMPLOYEE ID #:  818692173

ENTER PAY DATE AND GROSS AMOUNT FOR
ABOVE EMPLOYEE

PAY DATE: 850405      GROSS: 3000.00
PAY DATE: 850412      GROSS: 4000.00
PAY DATE: 850418      GROSS: 2500.00
```

When the operator presses Enter, the transaction processes. Processing continues until a line with no data is found or all lines are completed (whichever occurs first).

Using REPEAT to Display Multiple Records

You can display multiple segment instances on the screen by directing FIDEL to read and display the contents of a HOLD buffer. You can use a subscript value to identify a particular instance in the HOLD buffer with the following syntax

```
field(n)
```

where:

field

Is the name of a previously held field.

(n)

Is the integer subscript that identifies the number of the instance in the HOLD buffer containing the field to be displayed. *n* must be in integer format or the report group will be ignored.

The variable SCREENINDEX allows you to display and modify selected groups of records from the HOLD buffer.

Consider the following example, which uses the REPEAT statement to retrieve up to a set number (in this case, six) of multiple instances, saves them in the HOLD buffer, and then displays the instances on the CRTFORM:

```
MODIFY FILE EMPLOYEE
1.  CRTFORM
       "PAY HISTORY UPDATE"
       " "
       "ENTER EMPLOYEE ID#: <EMP_ID"
    MATCH EMP_ID
       ON NOMATCH REJECT
       ON MATCH GOTO COLLECT

    CASE COLLECT
2.  REPEAT 6 TIMES
```

```
2.     NEXT PAY_DATE
2.        ON NEXT HOLD PAY_DATE GROSS
3.        ON NONEXT GOTO DISPLAY

3.  ENDREPEAT
    GOTO DISPLAY
    ENDCASE
```

```
CASE DISPLAY
    IF HOLDCOUNT EQ 0 GOTO TOP;
4.  COMPUTE
       BUFFNUMBER/I5 = HOLDCOUNT;
5.  CRTFORM LINE 5
       "FILL IN GROSS AMOUNT FOR EACH PAY DATE"
       " "
       "PAY DATE          GROSS AMOUNT"
       "--------          ------------"
       "<D.PAY_DATE(1)    <T.GROSS(1)>"
       "<D.PAY_DATE(2)    <T.GROSS(2)>"
       "<D.PAY_DATE(3)    <T.GROSS(3)>"
       "<D.PAY_DATE(4)    <T.GROSS(4)>"
       "<D.PAY_DATE(5)    <T.GROSS(5)>"
       "<D.PAY_DATE(6)    <T.GROSS(6)>"
    GOTO UPDATE
    ENDCASE

    CASE UPDATE
6.  REPEAT BUFFNUMBER
       MATCH PAY_DATE
          ON NOMATCH REJECT
          ON MATCH UPDATE GROSS
    ENDREPEAT
    GOTO COLLECT
    ENDCASE
    DATA
    END
```

The procedure processes as follows:

1. When the procedure is executed, the first CRTFORM is displayed:

```
PAY HISTORY UPDATE

ENTER EMPLOYEE ID #:
```

2. Assume the operator enters the following ID and transmits the data:

```
ENTER EMPLOYEE ID #: 071382660
```

   If there is a match, the instruction is to REPEAT the logic six times. That is, up until six times, find a PAY_DATE and hold the PAY_DATE and the GROSS in the HOLD buffer.

3. When there are no more PAY_DATE fields or six of them have been held, the procedure branches to CASE DISPLAY.
4. The procedure stores the number of records that are in the HOLD buffer in the variable BUFFNUMBER.
5. The procedure displays the following CRTFORM:

```
PAY HISTORY UPDATE

ENTER EMPLOYEE ID #: 071382660

FILL IN GROSS AMOUNT FOR EACH PAY DATE

PAY DATE           GROSS AMOUNT
820831             916.67
820730             916.67
820630             916.67
820528             916.67
820430             916.67
820331             916.67
```

   The operator makes changes to the fields in the GROSS AMOUNT column and presses

Enter. All changes for all records are transmitted simultaneously as shown:

```
PAY HISTORY UPDATE

ENTER EMPLOYEE ID #: 071382660

FILL IN GROSS AMOUNT FOR EACH PAY DATE

PAY DATE           GROSS AMOUNT
820831             816.67
820730             816.67
820630             816.67
820528             916.67
820430             916.67
820331             916.67
```

6. The REPEAT statement instructs FOCUS to perform the MODIFY logic on all segment instances.

Note: If a CRTFORM screen with subscripted variables is rejected with a FORMAT ERROR, you may not alter any records on the screen prior to the record rejected, as FOCUS has already held them.

Using Groups of Fields With Case Logic

When you use Case Logic to process a group of fields, some important rules apply:

- Each time the procedure enters the case, the next group of fields is processed. FOCUS keeps track internally of which groups have been processed.
- If the CRTFORM with the group of fields is not in the TOP case, you must create your own branching logic to process all the groups before going back to the TOP. This normally requires some kind of counting mechanism. Once the procedure goes back to the TOP case, all unprocessed data on the CRTFORM or in a lowercase is lost.

Case Logic, Groups, CURSORINDEX and VALIDATE

In the following example, Case Logic is used with groups of fields. The CURSORINDEX (see Describing the CRT Screen) is used in conjunction with a VALIDATE:

```
MODIFY FILE EMPLOYEE
1.  CRTFORM
       "EMPLOYEE SALARY AND DEPARTMENT UPDATE"
       " "
       "PRESS ENTER"
    GOTO COLLECT

    CASE COLLECT
2.  REPEAT 6 TIMES
       NEXT EMP_ID
          ON NEXT HOLD EMP_ID CURR_SAL DEPARTMENT
          ON NONEXT GOTO DISPLAY
    ENDREPEAT
    GOTO DISPLAY
    ENDCASE

    CASE DISPLAY
3.  IF HOLDCOUNT EQ O GOTO EXIT;
4.  COMPUTE
       BUFFNUMBER/I5 = HOLDCOUNT;
5.  CRTFORM LINE 7
       "EMPLOYEE             SALARY               DEPARTMENT"
       "--------             ------               ----------"
       "<D.EMP_ID(1)         <:AAA.T.CSAL(1)>     <:BBB.T.DPT(1)>"
       "<D.EMP_ID(2)         <:AAA.T.CSAL(2)>     <:BBB.T.DPT(2)>"
       "<D.EMP_ID(3)         <:AAA.T.CSAL(3)>     <:BBB.T.DPT(3)>"
       "<D.EMP_ID(4)         <:AAA.T.CSAL(4)>     <:BBB.T.DPT(4)>"
       "<D.EMP_ID(5)         <:AAA.T.CSAL(5)>     <:BBB.T.DPT(5)>"
       "<D.EMP_ID(6)         <:AAA.T.CSAL(6)>     <:BBB.T.DPT(6)>"
```

```
6.  REPEAT 6 TIMES
       COMPUTE
          CURSOR/A66 = ':AAA';
          CURSORINDEX/I5=REPEATCOUNT;
       VALIDATE
          SALTEST = IF CSAL GT 50000 THEN 0 ELSE 1;
          ON INVALID TYPE "SALARY MUST BE LESS THAN $50,000"
```

```
         ON INVALID GOTO DISPLAY
    ENDREPEAT
    GOTO UPDATE
    ENDCASE

    CASE UPDATE
7.  REPEAT BUFFNUMBER
       MATCH EMP_ID
          ON NOMATCH REJECT
          ON MATCH UPDATE CURR_SAL DEPARTMENT
    ENDREPEAT
    GOTO COLLECT
    ENDCASE
    DATA
    END
```

The example processes as follows:

1. The first CRTFORM requests the operator to press Enter without typing anything.
2. The REPEAT statement retrieves six employee IDs, salaries, and department assignments and places them in a buffer.
3. If there are no records in the buffer, the procedure terminates.
4. The COMPUTE statement stores the number of records in the buffer in the variable BUFFNUMBER.
5. The second CRTFORM retrieves the IDs, salaries, and department assignments from the buffer and displays them together on the screen. Note the field labels:
   - The label :AAA on the CURR_SAL (CSAL) field.
   - The label :BBB on the DEPARTMENT (DPT) field.

   Assume that the operator changes the values to the following:

```
EMPLOYEE SALARY AND DEPARTMENT UPDATE

PRESS ENTER

EMPLOYEE             SALARY               DEPARTMENT
--------             ------               ----------
```

```
071382660           35000.00             PRODUCTION
112847612           23200.00             MIS
117593129           75480.00             MIS
119265415           19500.00             PRODUCTION
119329144           39700.00             PRODUCTION
123764317           36862.00             PRODUCTION
```

6. The second REPEAT statement operates on each of the six records displayed by the second CRTFORM, in order of display, performing the following tasks:

   - Sets the CURSOR variable to the label :AAA.
   - Sets the CURSORINDEX variable to the number of the record it's processing (1 through 6).
   - Validates the CURR_SAL field value. If the CURR_SAL value is $50,000 or more, the procedure branches back to the beginning of Case DISPLAY. The procedure displays the second CRTFORM again, with the CURSOR and CURSORINDEX variables positioning the cursor on the invalid salary.

   In the example, the procedure positions the cursor on the third CURR_SAL value:

```
EMPLOYEE SALARY AND DEPARTMENT UPDATE

PRESS ENTER

EMPLOYEE          SALARY               DEPARTMENT
--------          ------               ----------
071382660       35000.00               PRODUCTION
112847612       23200.00               MIS
117593129       75480.00               MIS
119265415       19500.00               PRODUCTION
119329144       39700.00               PRODUCTION
123764317       36862.00               PRODUCTION

(FOC421)TRANS 2 REJECTED INVALID SALTEST
SALARY MUST BE LESS THAN $50,000
```

7. If all values are valid, the third REPEAT statement updates the employee's salary and department for each record in the buffer. The procedure then branches to Case COLLECT to update six more records in the data source.

Handling Errors

It is important to know how various errors are handled by FIDEL so that proper instructions can be given to terminal operators. The following errors can cause a transaction or screen of data to be rejected:

- A format error, caused by entering non-numeric data for a numeric field.
- A validation error, caused by entering an incoming value that failed a VALIDATE test coded in the MODIFY.
- A NOMATCH condition, caused by entering data for a key field that did not match any record in the data source.
- A DUPLICATE condition, caused by key field values that matched records on a data source.
- An ACCEPT error, caused by entering a value for a data source field that failed the ACCEPT test.

Note: Error messages are discussed in detail in Messages: TYPE, LOG, and HELPMESSAGE.

Handling Format Errors

If the operator enters a non-numeric character into a field defined as numeric, an error message is displayed and the screen is not processed (processing stops). The error message indicates the line number and field name in error and the cursor is automatically positioned on that field. Additionally, if the operator enters a value that fails an ACCEPT test for a field an error message is displayed and the screen is not processed. Any message specified for that field with the HELPMESSAGE attribute will also be displayed.

The operator can retype the data and press the Enter key to retransmit the screen. Alternatively, the operator may press the PF2 key to cancel the transaction. The error prevents anything on the screen from being processed. When the operator corrects the error and transmits the screen, processing resumes.

There are two exceptions to this rule. When there are repeating groups, all complete transactions up to the error will be processed. Also, in REPEAT/HOLD loops, the data prior to the format error may not be altered.

VALIDATE and CRTFORM Display Logic

When the operator enters a value that is invalid, the transaction is rejected and an error message is displayed. By default, control returns to the first CRTFORM in the TOP case. However, you can use an ON INVALID GOTO statement to transfer control to any other case in the request.

If the NOCLEAR or blank option in the CRTFORM statement (see Using FIDEL in MODIFY) is in effect, the screen will not be cleared. The operator can change the data in the offending transaction and retransmit the screen.

When you use validations, you can divide the tests into different cases and repeat a case if it fails the test. The advantage of this is that the operator can change the invalid data and retransmit the screen before other sections are processed. An ON INVALID TYPE phrase can be used to send an informative message to the operator on the screen. The following example shows the use of these options:

```
CASE TRY
CRTFORM
   EMPLOYEE ID #: <EMP_ID NAME: <LAST_NAME"
   "CURRENT SALARY: <CURR_SAL"
VALIDATE
   GOODSAL= CURR_SAL GT 10000 AND CURR_SAL LT 1000000;
   ON INVALID TYPE
   THE CURRENT SALARY CANNOT BE LARGER THAN 1000000 OR"
   "LESS THAN 10000"
   ON INVALID GOTO TRY
   .
   .
   .
```

All messages appear on the bottom four lines of the screen, unless you specify the TYPE option on the CRTFORM statement (see Using FIDEL in MODIFY).

Handling Errors With Repeating Groups

If old style repeating groups (those without subscripts) are present and there is an error, processing continues to the next transaction on the screen. This means that if the operator changes the offending transaction and retransmits the screen, the other transactions on the screen become duplicates. It is important when using repeating groups to reject duplicates and turn the duplicate message off (LOG DUPL MSG OFF).

Alternatively, avoid using VALIDATE with repeating groups. Use COMPUTE instead and branch to a case that displays the erroneous data in a lower portion of the screen.

The following is an example of this technique. A test field is computed in Case TEST, using DECODE. This test field checks that the department value is a valid one. If the operator inputs a department value that is invalid, control branches to a case that displays the erroneous data (CASE BADDPT).

```
    MODIFY FILE EMPLOYEE
1.  CRTFORM
       "FILL IN THE FOLLOWING CHART WITH THE SALARIES"
       "AND DEPARTMENT ASSIGNMENTS OF FOUR NEW EMPLOYEES"
       " "
       "           EMPLOYEE ID       DEPARTMENT       SALARY"
       "           -----------       ----------       ------"
       "PERSON 1   <EMP_ID           <DEPARTMENT      <CURR_SAL"
       "PERSON 2   <EMP_ID           <DEPARTMENT      <CURR_SAL"
       "PERSON 3   <EMP_ID           <DEPARTMENT      <CURR_SAL"
       "PERSON 4   <EMP_ID           <DEPARTMENT      <CURR_SAL"
       GOTTO TEST

2.  CASE TEST
    IF EMP_ID IS ' ' GOTO TOP;
    COMPUTE
       TEST/I1 = DECODE DEPARTMENT (MIS 1 PRODUCTION 1 ELSE 0);
    IF TEST IS 0 GOTO BADDEPT ELSE GOTO ADD;
    ENDCASE

3.  CASE ADD
    MATCH EMP_ID
       ON NOMATCH INCLUDE
       ON MATCH REJECT
    ENDCASE

4.  CASE BADDEPT
    COMPUTE
       XEMP/A9 = EMP_ID;
```

```
      XDEPT/A10 = DEPARTMENT;
   CRTFORM LINE 12
      "INVALID ENTRY: DEPARTMENT MUST BE MIS OR PRODUCTION"
      "CORRECT THE ENTRY BELOW"
      " "
      "EMPLOYEE ID:  <D.XEMP    DEPARTMENT:  <T.XDEPT"
   COMPUTE
      DEPARTMENT=XDEPT;
   GOTO TEST
   ENDCASE

   DATA
   END
```

The request processes as follows:

1. This is the first and TOP case, and contains a CRTFORM that displays four instances of repeating groups. Assume the operator fills in values and transmits the screen. Control transfers to Case TEST.
2. Case TEST contains a computed field that uses DECODE to make sure that the values that have been input for DEPARTMENT are either MIS or PRODUCTION. When a DEPARTMENT value does not match this list, TEST is returned a code of 0, in which case control transfers to Case BADDEPT.
3. Case BADDEPT first computes two fields, XEMP and XDEPT, to have the values of EMP_ID and DEPARTMENT at the time the error occurred. Next, BADDEPT displays a CRTFORM containing a message to the operator and the two computed fields. The XDEPT field, which contains the invalid DEPARTMENT value, is a turnaround field so that the operator can see the invalid value and change it. Next, the COMPUTE is reversed and the new values are returned to their respective fields. Control transfers back to Case TEST where the DEPARTMENT values will continue to be tested until they are all valid. At that point, control transfers to Case ADD.
4. Case ADD contains the MATCH logic necessary to include new employees into the EMPLOYEE data source. The transaction including all the repeating groups is processed at one time.

Rejecting NOMATCH or Duplicate Data

When the request directs that transactions be rejected, an error message is displayed on the screen. It is a good idea, when the major keys do not repeat, to use a split CRTFORM

and give the keys in the first CRTFORM. Once the keys are accepted, the rest of the data may be entered. For example:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID#:     <EMP_ID"
   "THEN PRESS ENTER"
MATCH EMP_ID
   ON NOMATCH TYPE
      "ID NOT IN DATABASE   PLEASE REENTER"
   ON NOMATCH REJECT
   ON MATCH CRTFORM LINE 4
      "LAST NAME:   <T.LAST_NAME"
      "DEPARTMENT:  <T.DEPARTMENT"
      "SALARY:      <T.CURR_SAL"
   ON MATCH UPDATE LAST_NAME DEPARTMENT CURR_SAL
DATA
END
```

If the EMP_ID does not match, control returns immediately to the operator with a request to correct the value. If a match does occur, the operator must then fill in the balance of the form and transmit it.

If repeating groups are present and no other cases are involved, all of the groups are processed before control returns to the screen. Thus, splitting screens in this way is particularly useful when the second CRTFORM contains repeating groups.

Logging Transactions

You can log the data entered on the screen to any log file. Only the data is logged, not the CRTFORM, so a compact log file is created. For example:

```
LOG TRANS ON ALLDATA
```

This will log transactions to a file allocated to the ddname ALLDATA.

The record length of the file must allow space for each field on each CRTFORM in the procedure, plus one character at the start of each CRTFORM. The record length should not be longer than this.

This may be an inconvenient format, since it is very long if several CRTFORMs exist. Instead you can construct a custom log file of your own design using TYPE statements. This

example logs data collected from its preceding CRTFORM to a file allocated to ddname MYCRT, including a COMPUTE transaction number, TNUM:

```
CRTFORM
"EMPLOYEE ID #: <EMP_ID  NAME <LAST_NAME"
"HIRE DATE: <HIRE_DATE"
COMPUTE
TNUM/I4=TNUM+1;
TYPE ON MYCRT
"<TNUM><EMP_ID><LAST_NAME><HIRE_DATE"
```

This option is preferable to the standard LOG option whenever a procedure contains more than two CRTFORMs, or when text or computed fields must be captured on the log file.

Additional Screen Control Options

MODIFY CRTFORMs support several additional screen control options:

- Clearing the screen with CLEAR/NOCLEAR.
- Specifying the screen size with WIDTH/HEIGHT.
- Changing the size of the message area at the bottom of the screen using Using FIDEL in MODIFY. This increases the length of the screen that can be used for the actual form.

Clearing the Screen: CLEAR/NOCLEAR

Data is transmitted from the CRTFORM to the data source when the operator presses the Enter key. After each successful screen is processed, the data areas are automatically cleared. You can override this default by using the NOCLEAR option. Then, after each data transmission, the screen remains unchanged.

This is a useful feature when there is a substantial amount of data that carries over from one screen to another. The syntax is

```
CRTFORM action
```

where:

action

Is one of the following:

blank is the default. Causes the screen to clear after the data is transmitted. If a transaction is invalid and an error message appears at the bottom of the screen, the screen will not be cleared.

NOCLEAR causes the data values on the screen to remain as is after data is transmitted.

CLEAR causes the data values on the screen to clear after every data transmission, even if there is an error. Thus, if CLEAR is specifically used and there is an error, data must be reentered.

Note: The options chosen may be different from one CRTFORM to the next.

Specifying Screen Size: WIDTH/HEIGHT

FIDEL assumes a default screen size of 24 lines of 80 characters each. The WIDTH/HEIGHT options allow you to use the full width and height of IBM terminals that are larger than the usual 3270 screen for the display of CRTFORMs. The following syntax allows you to override the defaults

```
CRTFORM [WIDTH nnn] [HEIGHT nnn]
```

where:

WIDTH nnn

Is the total number of characters across the face of the screen. Acceptable values for WIDTH are 80 and 132 and cannot exceed the true width of the terminal. FOCUS verifies that each line of the CRTFORM can be displayed at the current WIDTH specification. If any line of the CRTFORM exceeds it, you will receive error message FOC456, and the procedure will not run.

HEIGHT nnn

Is the total number of lines that each screen supports. It bears no relation to the number of lines in the CRTFORM. It may not exceed the true height of the terminal, but it may be less. For example, you can specify HEIGHT 20 for a Model 2 screen instead of 24 and write a CRTFORM of 32 lines. The first 16 lines appear on one screen and the next 16 on the subsequent screen. Remember that by default, four lines are reserved for

TYPE messages.

The following table gives the physical screen sizes for the IBM 3270 series of terminals:

Terminal Type	Model	Width	Height
3270	1	80	24
3277, 3278, 3279, 3178	2	80	24
3278, 3279	3	80	32
3278	4	80	43
3278	5	132	27

FOCUS senses the width and height of the terminal which you are using and attempts to implement your CRTFORM WIDTH and HEIGHT specifications accordingly. Here are some rules and facts that apply:

- If your WIDTH or HEIGHT specifications exceed the perceived characteristics of the terminal, you will receive a FOC491 error message and the procedure will not run.
- FOCUS sees the terminal as it is defined to the operating system. For example, a Model 5 3278 may be defined to the operating system as a Model 2 terminal. That terminal will appear to FOCUS as a Model 2 (24 lines deep and 80 characters wide). A WIDTH 132 specification will produce a FOC491 error message.

Changing the Size of the Message Area: TYPE

By default, FOCUS reserves the last four lines of the terminal screen for TYPE messages and text messages specified with the HELPMESSAGE attribute (see Messages: TYPE, LOG, and HELPMESSAGE). You can override this default and determine the number of lines each CRTFORM reserves with the keyword TYPE. This feature allows you to increase the number of lines on the screen for CRTFORM display and reduce the number of lines reserved for messages at the bottom of the screen. The syntax is

```
CRTFORM TYPE {n|4}
```

where:

n

Is a number from one to four indicating the number of message lines desired. The TYPE value setting remains in effect for all subsequent CRTFORMs in the same procedure until overridden by a new value.

You can expand the actual CRTFORM screen size by specifying a number less than four. For example, a terminal with a height of 24 lines currently reserves 20 lines for the CRTFORM and four lines for the TYPE area. If you specify a TYPE area of 2, the CRTFORM area increases to 22 lines.

If one procedure varies the size of the TYPE area from a larger to a smaller number, CRTFORM will clear the necessary TYPE statements in order to generate the next screen. If multiple CRTFORMs are written to the same screen, each CRTFORM should specify the same TYPE area size. For example:

```
CRTFORM LINE 1 TYPE 2
:
:
CRTFORM LINE 7 TYPE 2
```

Messages supplied with the HELPMESSAGE attribute in the Master File for fields on the MODIFY CRTFORM, are displayed in the TYPE area.

This type of message consists of one line of text which is displayed when:

- The value entered for a data source field is invalid according to the ACCEPT test for the field, or causes a format error.
- The user places the cursor in the data entry area for a particular field and presses a predefined PF key. If no message has been specified for that field, the following message will be displayed:

  ```
  NO HELP AVAILABLE FOR THIS FIELD
  ```

Using FIDEL in Dialogue Manager

FIDEL works with all the standard Dialogue Manager facilities. However, the following differences apply when you use FIDEL with Dialogue Manager:

- You must allocate space for the variable field on the -CRTFORM, because variable fields in Dialogue Manager are not related to a Master File (see Using FIDEL in Dialogue Manager).
- There are two additional control statements: -CRTFORM BEGIN and -CRTFORM END. These give you control over when you begin and end the form (see Starting and Ending CRTFORMS: BEGIN/END). This control allows you to make use of other Dialogue Manager control statements as you are building your -CRTFORM.

Allocating Space on the Screen for Variable Fields

You must define the length of variable fields in -CRTFORMs. The length of Dialogue Manager variables can be defined in one of two ways:

- Directly on the -CRTFORM using the following syntax for allocating space.

  ```
  <&variable/length
  ```

  where:

  length

  Is a number representing the alphanumeric length of the variable.

- By using the -SET command to pre-declare the allocation of space using the syntax

  ```
  -SET &variable = ' ' ;
  ```

  where:

  ' '

  Represents the alphanumeric length of the variable.

  Note:

  - If the variable format has been previously defined in the FOCEXEC procedure, the length defined directly on the -CRTFORM supersedes the previously defined format permanently.
  - Variables used as label names (&:variable) cannot be automatically defined on

the -CRTFORM. These variables must be defined with -SET statements.

Starting and Ending CRTFORMS: BEGIN/END

-CRTFORM BEGIN indicates that the form is being built. This Dialogue Manager control statement enables you to use other Dialogue Manager control statements between the screen lines without causing the CRTFORM to end. This is necessary when you are using indexed variables in a looping procedure.

-CRTFORM END terminates the form and causes the display of the assembled form.

Using Indexed Variables With -CRTFORM BEGIN and -CRTFORM END

The following is an example of the use of indexed variables in -CRTFORM. The variable &LINENUM is the indexed variable in the -CRTFORM. The index, &I, is set to increment by 1 each time a line is written. After the 10th line, the -CRTFORM ends. Note the use of the Dialogue Manager label, -BUILD and the -SET statement to control the loop within the form:

```
1.  -SET &I = 0;
2.  -CRTFORM BEGIN
    -"THE FOLLOWING FORM STORES 10 LINES OF TEXT"
    -" "
3.  -BUILD
4.  -SET &I = &I + 1;
5.  -SET &LINENUM.&I = 'LINE ' | &I;
6.  -"<D.&LINENUM.&I <&LINE.&I/60"
7.  -IF &I LT 10 GOTO BUILD;
8.  -CRTFORM END
    -*
    -TYPE LINE #2 CONTAINS THE FOLLOWING TEXT:
    -TYPE
9.  -TYPE &LINE2
```

This example processes as follows:

1. This -SET statement declares a counter, &I, and sets the counter to 0.

2. The -CRTFORM BEGIN statement begins the form.
3. This statement is a Dialogue Manager label, -BUILD. Because we are using the -CRTFORM BEGIN statement, this label does not end the CRTFORM.
4. This -SET statement sets the counter &I to increment by 1 each time a line is written. This controls the loop within the form.
5. This -SET statement indexes the variable &LINENUM with the counter &I. Thus, each time it is encountered in the -CRTFORM it will increment +1.
6. The -CRTFORM will appear as follows:

```
THE FOLLOWING FORM STORES 10 LINES OF TEXT

LINE 1
LINE 2
LINE 3
LINE 4
LINE 5
LINE 6
LINE 7
LINE 8
LINE 9
LINE 10
```

   Type any text you wish onto the lines.
7. The -IF test allows the loop to process until there are 10 lines in the -CRTFORM. At that point control transfers to the -CRTFORM END statement.
8. -CRTFORM END ends the -CRTFORM and causes it to be displayed.
9. The last TYPE statement shows the contents of line 2.

Clearing the Screen in Dialogue Manager

The statement -CRTFORM both initiates the screen form and automatically clears the screen. The screen form begins at the top of the screen.

After the operator enters values for the variables and presses Enter, the variables are supplied with the values and the screen is cleared.

Changing the Size of the Message Area: -CRTFORM TYPE

By default, FOCUS reserves the last four lines of the Dialogue Manager terminal screen for TYPE messages. You can change this by using the keyword TYPE to determine the number of lines each CRTFORM reserves for messages. This feature allows you to increase the number of lines on the screen for CRTFORM display and reduce the number of lines reserved for messages at the bottom of the screen. The syntax is

```
-CRTFORM TYPE {n|4}
```

where:

n

Is a number from 1 to 4 indicating the number of message lines desired. The TYPE value setting remains in effect for all subsequent CRTFORMs in the same procedure until overridden by a new value. The default is 4.

You can expand the CRTFORM screen size by specifying a number less than 4. For example, a terminal with a height of 24 lines reserves 20 lines for the CRTFORM and four lines for the TYPE area. If you specify a TYPE area of 2, the CRTFORM area increases to 22 lines.

Annotated Example: -CRTFORM

The following FOCEXEC is an example of a TABLE request incorporating the use of -CRTFORM.

```
    -* Component Of Retail Sales Reporting Module
1.  SET &LIST = 'STAMFORD,UNIONDALE,NEWARK';
2.  PROMPT &CITY.(&LIST).ENTER CITY.:
    -*
3.  -CRTFORM
    -"Monthly Sales Report For <D.&CITY"
    -"Date: <D.&DATE     Time: <D.&TOD"
    -" "
    -"Beginning Product Code is:  <&BEGCODE/3"
    -"Ending Product Code is:     <&ENDCODE/3"
    -"Regional Supervisor is:     <&REGIONMGR/15"
```

```
    -"Title For UNIT_SOLD is:     <&UNIT_HEAD/10"

4.  TABLE FILE SALES
    HEADING CENTER
    MONTHLY REPORT FOR &CITY"
    "PRODUCT CODES FROM &BEGCODE TO &ENDCODE"
    SUM UNIT_SOLD AS &UNIT_HEAD
    AND RETURNS AND COMPUTE
    RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
    BY PROD_CODE
    IF PROD_CODE IS-FROM &BEGCODE TO &ENDCODE
    IF CITY EQ &CITY
    FOOTING CENTER
    "REGION MANAGER: &REGIONMGR"
    "CALCULATED AS OF &DATE"
5.  END
```

The following is a sample of the dialogue between the screen and the operator. Operator entries are in lowercase.

1. The -SET statement sets a value for the variable &LIST. The value is actually a list of the names of three cities. They are enclosed in single quotation marks because of the embedded commas.
2. The -PROMPT statement prompts the operator at the terminal for a value for &CITY. Assume the operator types a city that is not on the list:

```
ENTER CITY:
boston
PLEASE CHOOSE ONE OF THE FOLLOWING:
STAMFORD,UNIONDALE,NEWARK
ENTER CITY:
stamford
```

3. The statement -CRTFORM initiates a screen form on which you type data:

```
Monthly Sales Report for STAMFORD
Date: 01/08/2003              Time: 13.12.41
```

```
Beginning Product Code is:     b10
Ending Product Code is:        b20
Regional Supervisor is:        smith
Title For UNIT_SOLD is:        sales
```

4. The following are the stacked FOCUS commands as they appear on the FOCSTACK after the values have been entered from the -CRTFORM:

```
TABLE FILE SALES
HEADING CENTER
"MONTHLY REPORT FOR STAMFORD"
"PRODUCT CODES FROM B10 TO B20"
" "
SUM UNIT_SOLD AS SALES AND RETURNS AND COMPUTE
RATIO/D5.2 = 100 * RETURNS/UNIT_SOLD;
BY PROD_CODE
IF PROD_CODE IS-FROM B10 TO B20
IF CITY EQ STAMFORD
FOOTING CENTER
"REGION MANAGER: SMITH"
"CALCULATED AS OF 01/08/2003"
END
```

5. The report is as follows:

```
PAGE     1

  MONTHLY REPORT FOR STAMFORD
 PRODUCT CODES FROM B10 TO B20

PROD_CODE  SALES   RETURNS  RATIO
---------  ------  -------  -----
B10            60       10  16.67
B12            40        3   7.50
B17            29        2   6.90

             REGION MANAGER: SMITH
              CALCULATED AS OF 11/04/03
```

Using the ibi FOCUS Screen Painter

The FOCUS Screen Painter allows you to design a FIDEL full-screen layout by placing literal text and areas for fields on the screen in any position that you desire. You then assign these field areas of the screen to a data source or computed fields, and FOCUS automatically codes the CRTFORM. You can also color, highlight, and/or assign screen attributes to sections of the screen (text, fields, background or any combination).

The FOCUS Screen Painter also allows you to generate CRTFORMs automatically without specifying field names (see Using FIDEL in MODIFY).

The Screen Painter operates within TED, the FOCUS editor (see the *Overview and Operating Environments* manual for more details on TED), and can be used to create both MODIFY CRTFORMs and Dialogue Manager -CRTFORMs. It is easy to use and makes the creating of forms simple and visual.

Entering Screen Painter

To create a CRTFORM using the Screen Painter, you first enter the PAINT command from within TED. You can set up the PAINT screen as follows:

1. Enter TED by typing TED followed by the name of the file:

   ```
   TED FOCEXEC(CRTEMP
   ```

   This opens the FOCEXEC called CRTEMP. The FOCEXEC may or may not already exist.

2. Place a CRTFORM or -CRTFORM statement in the FOCEXEC if it is not already there. For example:

   ```
   MODIFY FILE EMPLOYEE
   CRTFORM
   ```

3. When a FOCEXEC is on the screen, enter the PAINT command in the command area or press PF4. TED searches from the current line down the file until it finds a CRTFORM statement and makes the following line the current line. (If you use more than one CRTFORM in the FOCEXEC and you want to create the second CRTFORM, enter the command PAINT 2.)

   Note: A Master File must be active for the Screen Painter to set the default field

lengths for data source fields.

The following PAINT screen is displayed on your terminal:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7..+...









...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
COMMAND:_

01=HELP 03=END 07=BACKWARD 08=FORWARD 09=ASGN-FLD 10=ASSIGN
11=FIDEL 17=BOX
```

Between the two scale lines are 20 blank lines in which to enter the screen layout. The cursor is positioned in the command zone in the lower left portion of the screen. The codes at the bottom of the screen identify some of the PF keys that you can use.

These perform the following functions:

PF Key	Function
01=HELP	Lists all the PF key functions.
03=END	Transfers you from the PAINT screen back into TED, within your file.
07=BACKWARD	Scrolls back to the previous screen of the CRTFORM. When used with ASSIGN, moves the cursor back to the first field.

PF Key	Function
08=FORWARD	Scrolls forward to the next screen of the CRTFORM. When used with ASSIGN, moves the cursor to the next field.
09=ASGN-FLD	Use on the ASSIGN screen. Transfers you to the particular field that the cursor is placed on. You can then immediately assign or change attributes for that field.
10=ASSIGN	Transfers you from the PAINT screen to the ASSIGN screen (see Identifying Fields: ASSIGN).
11=FIDEL	Shows you the CRTFORM as it will appear on the screen.
17=BOX	Enables you to define a box of text. Move the cursor to the upper-left corner and press PF17. Select features from the box menu and then move the cursor to the bottom-right corner and press PF17.

Note: With the exception of FORWARD, BACKWARD and ASGN-FLD, you can also accomplish these functions by typing the command name in the command zone.

4. If the CRTFORM already includes fields, and one or more fields are not declared in the Master File, you may see this message:

```
(FOC532) LENGTHS OF FIELDS IN THIS CRTFORM CANNOT BE DETERMINED
```

To continue type IGNore and provide the lengths explicitly, or type ?F filename to activate the appropriate master. After you follow the message instructions, the PAINT screen appears.

PF Keys in PAINT

You can alter the values of PF keys in PAINT with the command

```
SET PFnnword
```

where:

nn

Is a number from 1 to 24 specifying the PF key to be set.

word

Is the new value for the key.

The initial PF key settings in PAINT are:

PF Key	Setting
PF1, PF13	: HELP
PF2, PF14	: INSERT
PF3, PF15	: END
PF4	: PAINT
PF5	: TOP
PF6	: BOTTOM
PF7, PF19	: BACKWARD PAGE
PF8, PF20	: FORWARD PAGE
PF9	: ASSIGN FIELD
PF10	: ASSIGN
PF11	: FIDEL
PF12	: DUPLICATE
PF16	: QUIT
PF17	: BOX
PF18	: (currently not used)

PF Key	Setting
PF21	: CRTFORM
PF22	: SET OUTPUT FIDEL
PF23	: SET OUTPUT DIALOGUE
PF24	: (currently not used)

Entering Data Onto the Screen

In PAINT, you may enter text, and specify field dimensions. Always use the arrow keys to designate text and field areas on this screen. Generally, text is entered by positioning the cursor and typing, but fields require type and width specifications.

To create a field, type

```
<xx...x
```

where the total number of x's equals the width of the field desired. If you do not specify a width, or if the command you entered is not syntactically correct, or active, PAINT will automatically default to a width defined in the Master File.

Fields are conditional by default. To specify non-conditional fields, enter

```
<xx...x>
```

where the total number of x's equals the width of the field.

You may enter text descriptions of each field, but do not type the field name after the left or right caret. Later you will learn how to assign each field a field name. You may designate the field as Entry, Turnaround or Display with the ASSIGN command (see Identifying Fields: ASSIGN). By default, the fields are conditional. To specify non-conditional, type a right caret (>) after the x's that indicate the field. We recommend that turnaround fields be non-conditional. (See Conditional and Non-Conditional Fields for information on conditional and non-conditional fields.)

Editing Functions

When you are designing your screen, you have editing functions available to you. To use them, you must enter the command name on the COMMAND line on your PAINT screen or use the appropriate PF key:

- Inserting Lines: INSERT, PF2, PF14. You can insert lines by moving the cursor to any character on a line. Press PF2 or PF14 and the new line will be inserted immediately following the line where the cursor is positioned. If you want to insert more than one line, type the command (do not press Enter)

  ```
  I[NSERT] n
  ```

  where *n* is the number of new lines to be inserted. Next, move the cursor to the line where you want the lines inserted. Press Enter and *n* lines will be inserted beneath the line where the cursor is currently positioned.

  If the insert causes the screen to exceed 20 lines, the message

  ```
  1,40
  ```

  will be displayed, indicating that the display starts at line 1 out of a total of 40.

- Deleting Lines: DELETE. You can similarly delete lines by typing:

  ```
  D[ELETE] n
  ```

  on the command line, where *n* is the number of lines you want deleted. Next, move the cursor to the first line you want deleted and press Enter.

- Duplicating Lines: DUPLICATE, PF12. You can duplicate lines by placing the cursor on the line that you want to duplicate. Press PF12. If you want to duplicate more than one line, type the command

  ```
  DU[PLICATE] n
  ```

  where *n* is the number of copies you want; position the cursor on the line you want to duplicate and press Enter.

  If the line that you are copying contains subscripted fields (for example, "SALES (1)"), the subscripts will be incremented by one automatically (see Using FIDEL in MODIFY). If you want an increment other than 1, enter the command

```
DUPLICATE n m
```

where *m* is the increment number.

Sample PAINT Screen

In the following example, assume that the following FOCEXEC exists:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID #: <EMP_ID"
MATCH EMP_ID
   ON NOMATCH REJECT
   ON MATCH CRTFORM
```

To use the Screen Painter to create the second CRTFORM, specify PAINT 2 at the TED command line (2 indicates second CRTFORM). Then type the following text and fields on the PAINT screen to create the CRTFORM that will be displayed if there is a match on EMP_ID.

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

                                EMPLOYEE UPDATE

EMPLOYEE ID #: <XXXXXXXXX            LAST NAME: <XXXXXXXXXXXXXXX

DEPARTMENT: <XXXXXXXXXX>             CURRENT SALARY: <XXXXXXXX

BANK: <XXXXXXXXXXXXXXXXXXXX




...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
COMMAND:_

01=HELP 03=END 07=BACKWARD 08=FORWARD 09=ASGN-FLD 10=ASSIGN 11=FIDEL
17=BOX
```

When you finish entering text and indicating areas for fields (the number of X's corresponds to the field length), press **Enter**. The following screen results:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

                                EMPLOYEE UPDATE

EMPLOYEE ID #: <111111111           LAST NAME: <22222222222222

DEPARTMENT: <1111111111>            CURRENT SALARY: <22222222

BANK: <11111111111111111111




...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
COMMAND:_

01=HELP 03=END 07=BACKWARD 08=FORWARD 09=ASGN-FLD 10=ASSIGN 11=FIDEL
17=BOX
```

Note that the X's are replaced with numbers indicating the relative position of each field on a line. On the second line, EMPLOYEE ID is number 1 and LAST NAME is number 2.

Note: Labels created in Screen Painter cannot exceed 12 characters.

Defining a Box on the Screen

You can define a boxed area of the screen, have it flash, or underline it. Text within the box assumes the attributes of the box, but fields within the box do not change their appearance.

To define a box, place the cursor in the upper-left corner of the area you want to enclose in a box, and press PF17. The following screen and menu appear:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
```

```
                                EMPLOYEE UPDATE

EMPLOYEE ID #: <111111111           LAST NAME: <22222222222222

DEPARTMENT: <1111111111>            CURRENT SALARY: <22222222

BANK: <11111111111111111111




...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
Color (W,B,R,P,G,A,Y):           Flash /Under/Inv/Off (F,U,I,O):
Please position the cursor at other end of box and hit the key again
```

Fill in the color and/or attributes that you desire, position the cursor at the lower-right corner of the area you want to enclose in a box, and press PF17.

To delete the box, move the cursor to the upper-left corner of the box and type O in the attribute area. Then move the cursor to the lower-right corner of the box and press PF17. The letter O stands for OFF and deletes the box. Note that you must position the cursor exactly at the corners.

The BOX feature of Screen Painter will not generate a proper box if the fields cross or touch the boundary of the box itself. Boxes may not extend past column 77.

If you try to generate a box, but fail, the following message appears:

```
command.box
(FOC694) INVALID BOX REGION OR CURSOR POSITION DEFINED.
```

When this happens, press Enter to clear the message, move the cursor to the upper-left corner, and press PF17 to start over.

If you press PF17 to begin a box and then decide not to define a box, press PF3 to cancel.

Identifying Fields: ASSIGN

Until now, you have simply laid out text that describes the fields, designated a display length (X's) within the left caret (<), and possibly indicated non-conditional (>) fields. Now

you can assign field names and attributes for the fields. Enter the command ASSIGN in the command zone or press PF10. Your ASSIGN screen displays the following:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

                                EMPLOYEE UPDATE

EMPLOYEE ID #: *********            LAST NAME: EEEEEEEEEEEEEE

DEPARTMENT: EEEEEEEEEE             CURRENT SALARY: EEEEEEEE

BANK: EEEEEEEEEEEEEEEEEEE




...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
Field:               Entry/Turn Disp (E,T,D):    Col (W,B,R,P,G,A,Y):
Field Length: 9(D12.2M)  High/Nodis/Inv (H,N,I):     Label:
```

The first field following the descriptive text EMPLOYEE ID #: is highlighted and replaced by asterisks. All other fields are displayed in low intensity with E's denoting the length of the fields. The cursor is positioned in the status entry area at the bottom of the screen next to FIELD.

Now you can enter and assign field names and attributes for the field appearing in asterisks. Fill in the appropriate values in the status entry area at the bottom of the screen. To move from one status area to the next, press TAB. You may leave a blank where you do not want to use a particular attribute.

FIELD:

Enter the field name for the first field. In this case, enter EMP_ID, which is the name of the field in the Master File.

ENTRY/TURN/DISP (E,T,D):

You may designate the field as Entry, Turnaround, or Display by specifying E, T, or D, respectively. The default is Entry. (See Data Entry, Display and Turnaround Fields for more information on Entry, Turnaround, and Display fields.) You specify whether a field is conditional or non-conditional when you enter the field on the PAINT screen (see Using the ibi FOCUS Screen Painter).

COL (W,B,R,P,G,A,Y):

You may designate the field with a color by entering one of the color abbreviations in the COL area. You may choose W, white; B, blue; R, red; P, pink; G, green; A, aqua; Y, yellow. If you do not wish to assign a color, leave this area blank.

FIELD LENGTH: 9 (A9):

In MODIFY, if a Master File is active while you are assigning attributes, the LENGTH status will contain two values: the first value is the number of X's from the PAINT screen, which is the display value; the value in parentheses is the format value from the Master File. The display value must be equal to or less than the format value.

If you want to change the display value on the screen, put a new number in the FIELD LENGTH area or return to PAINT (PF3) and enter the correct number of characters following the <.

HIGH/NODISP/INV (H,N,I):

You can choose highlight, nodisplay or inverse video as an attribute for the field by filling in the appropriate abbreviation.

LABEL:

If you want to enter a label, simply enter its name. The colon and period are automatically provided on the screen.

In the following example, the current field is LAST_NAME. It is designated a display field. The remaining attributes are left blank. After you press Enter and move to the next field, the asterisks turn to D's (display) as did the EMP_ID field.

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

                                EMPLOYEE UPDATE

EMPLOYEE ID #: DDDDDDDDD           LAST NAME: ***************

DEPARTMENT: EEEEEEEEEE             CURRENT SALARY: EEEEEEEEEEEEEE

BANK: EEEEEEEEEEEEEEEEEEE
```

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
FIELD: last_name      ENTRY/TURN/DISP (E,T,D): d    COL (B,R,P,G,A,Y):
FIELD LENGTH: 15 (A,15) HIGH/NODISP/INV (H,N,I):       LABEL:
```

To move to the next field, press PF8. You may assign a field name, prefix, color, attribute or label to the remaining fields on the screen. If you need to move to a previous field to change something, press PF7. This will return you to the first field. From there you can use the TAB key to move to the field that you need.

To move to a specific field directly from PAINT or from within ASSIGN, place the cursor on that field and press PF9, ASGN-FLD.

Viewing the Screen: FIDEL

From the PAINT or ASSIGN screen, you can view the exact FIDEL screen that you have created. Press PF11 or type FIDEL in the command zone. As the following screen shows, all entry fields are blank and ready to receive data; all turnaround fields contain T's and may be typed over; all display fields contain D's and are protected:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

                                EMPLOYEE UPDATE

EMPLOYEE ID #: DDDDDDDDD            LAST NAME: DDDDDDDDDDDDDDD

DEPARTMENT: TTTTTTTTTT              CURRENT SALARY:



FIDEL: Press PF3 or PF15 to return to the PAINT screen.
```

As indicated on the FIDEL screen, to return to the PAINT screen press PF3 or PF15.

Generating CRTFORMs Automatically

To generate CRTFORMs automatically (that is, without specifying individual fields) from the FOCUS Screen Painter, use the asterisk (*) with CRTFORM in the PAINT screen command zone. (See Using FIDEL in MODIFY for information on CRTFORM * variations and syntax.)

The text description identifying field is the field name from the Master File. Key fields automatically become entry fields, and all other fields become turnaround fields. With multi-segment data sources, the CRTFORM * command ignores all segments following the first cross-reference (segment type KU or KM) described in the Master File.

For example, to generate a CRTFORM containing all fields in the EMPLOYEE Master File, do the following:

1. Type a MODIFY and a CRTFORM statement in a FOCEXEC.
2. Enter PAINT on the TED command line to invoke the Screen Painter.
3. Type CRTFORM * in the Screen Painter command zone.

The following PAINT screen results:

```
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...

EMP_ID          :<111111111>   :
LAST_NAME       :<111111111111111   :        FIRST NAME  :<2222222222:
HIRE_DATE       :<111111    :                DEPARTMENT  :<2222222222:
CURR_SAL        :<111111111111  :            CUR_JOBCODE :<222   :
ED_HRS          :<111111    :
BANK_NAME       :<1111111111111111111  :
BANK_CODE       :<111111    :                BANK_ACCT   :<222222222:
EFFECT_DATE     :<111111    :
DAT_INC         :<111111>   :
PCT_INC         :<111111    :                SALARY      :<222222222222:
JOBCODE         :<111      :
TYPE            :<1111>    :
ADDRESS_LN1     :<1111111111111111111  :
ADDRESS_LN2     :<1111111111111111111  :
ADDRESS_LN3     :<1111111111111111111  :
ACCTNUMBER      :<111111111  :
PAY_DATE        :<111111>   :
GROSS           :<111111111111   :
DED_CODE        :<1111>   :
```

```
PF8=NEXT SCREEN PF7=PREVIOUS SCREEN PF1=OUT




...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...
COMMAND:                                                  1, 40
01=HELP 03=END 07=BACKWARD 08=FORWARD 09=ASGN-FLD 10=ASSIGN 11=FIDEL
17=BOX
```

CRTFORM * creates labels (that is, text describing each field) on the CRTFORM of up to 12 characters. If the field name is shorter than 12 characters, the label is the field name. If the field name exceeds 12 characters, a caret (>) in the 12th position indicates a longer field name.

Terminating Screen Painter

To return to TED from the PAINT screen, enter the command END in the command zone or press PF3 until the prompt for TED appears. TED displays the lines as they have been generated, beginning at the current line, which is ON MATCH CRTFORM:

```
" <.C.                    EMPLOYEE UPDATE                          <0X
" <.C.                                                             <0X
                        <.C."
" <.C. EMPLOYEE ID #: <D.EMP_ID/09                  LAST NAME:     <0X
<LAST_NAME/15           <.C."
" <.C.                                                             <0X
                        <.C."
" <.C. DEPARTMENT: <T.DEPARTMENT/10>           CURRENT SALARY:     <0X
<T.CURR_SAL/08          <.C."
" <.C.                                                             <0X
                        <.C."
" <.C. BANK <T.BANK_NAME/20                                      <.C."
" <.C.                                                             <0X
DATA
END
```

The generated code for the CRTFORM is in the file. Notice that each field is named and has its length appended to it. Any attributes or labels requested during the ASSIGN process are also present. If you want to change the layout, you can use the TED editor or you can return to the PAINT and/or ASSIGN screen to make the changes.

You can add further MATCH logic to the FOCEXEC by using TED. For example:

```
MODIFY FILE EMPLOYEE
CRTFORM
   "ENTER EMPLOYEE ID #: <EMP_ID"
   MATCH EMP_ID
      ON NOMATCH REJECT
      ON MATCH CRTFORM
"        EMPLOYEE UPDATE"
" "
" EMPLOYEE ID #: <D.EMP_ID/09            LAST NAME: <D.LAST_NAME/15"
" "
" DEPARTMENT: <:FIRST.H.T.DEPARTMENT/10> CURRENT SALARY: <0X
<.C.CURR_SAL/08"
" "
" BANK : <BANK_NAME/20"
      ON MATCH UPDATE DEPARTMENT CURR_SAL
      ON MATCH CONTINUE TO BANK_NAME
      ON NOMATCH INCLUDE
      ON MATCH REJECT
DATA
END
```

If you want to add another CRTFORM screen at this point, make sure you are on the current line, type the CRTFORM or -CRTFORM statement, and reenter PAINT to design the next screen. Finally, you can exit the PAINT screen, return to TED, and add or change further logic.

Alternatively, all of the logic of the request could have been entered first and then the Screen Painter used to create all the FIDEL screens. To create the first screen, enter the command PAINT or PAINT 1; to create the second screen, enter the command PAINT 2. PAINT 2 locates the second CRTFORM statement starting from the current line. You can continue with PAINT 3, and so on, for all subsequent CRTFORM statements in the procedure.

Directly Editing ibi FOCUS Databases With FSCAN

The full-screen FSCAN facility enables you to edit FOCUS databases directly on your terminal screen. You can use FSCAN to add, update, and delete data from FOCUS databases as if the segments in the FOCUS databases were flat files on a full-screen editor. You can type over field values, or change them by issuing commands.

Introduction

FSCAN enables you to:

- Add records to new or existing FOCUS databases.
- Change field values in FOCUS databases. With FSCAN you can change the values in key fields (not possible with MODIFY requests).
- Delete records from FOCUS databases.
- Search through FOCUS databases to locate instances of specified character strings or values.

If your database is protected by shadow paging, the changes you make on FSCAN are not permanent until you issue a command to do so. You may choose to exit FSCAN without saving any of the changes.

Databases on Which FSCAN Can Operate

FSCAN can operate on databases having the following attributes:

- The databases are FOCUS databases, not databases of other types.
- The databases are individual databases, not combined structures created by the COMBINE command.

- The length of the root key field in the database does not exceed 61 bytes, and the sum of the field name length plus the field length does not exceed 73 bytes.

Also, note the following regarding databases:

- FSCAN does not accept alternate file views.
- Databases that you specify with the USE command using the READ option are write protected.
- Databases that you are viewing on a FOCUS Database Server in Simultaneous Usage mode are write protected.

Segments on Which FSCAN Can Operate

The following rules apply to the display and editing of segments in FSCAN:

- FSCAN does not display a segment containing a key field longer than 61 bytes and the sum of the field name length plus the field length does not exceed 73 bytes, nor does it display the descendants of that segment.
- When you input a new segment instance, the instance must have a key unique to its group. (In the root segment, this means all the instances in the segment; in a descendant segment, this means all the instances that share a parent instance). If you try to input an instance with a duplicate key, FSCAN will generate an error message.
- If you change a key field value of an instance, the new instance key (the combination of all key field values in the instance) must be unique to the group. If you try to change the key to a duplicate, FSCAN will generate an error message.
- If you use FSCAN on segments already containing duplicate keys, the results are unpredictable. If the root segment has duplicate keys, an attempt to display a screen with these duplicates results in FSCAN terminating in an error. If a descendant segment has duplicate keys, an FSCAN error is displayed and you are positioned at the parent segment.
- When a segment is type S0 or blank, no one field is designated as the key field. FSCAN considers all fields in such segments to be key fields. This has two ramifications:
    - You cannot input a segment instance that is the duplicate of another in the

same group.

- You cannot update a segment instance so that it duplicates another segment instance in the same group.

Fields That FSCAN Can Display

FSCAN can display fields containing the following attributes:

- The field length does not exceed 61 bytes and the sum of the field name length plus the field length does not exceed 73 bytes.
- The fields are real database fields, not DEFINEd fields.
- FSCAN displays group fields as their individual members, not as a group.

Note: Text fields cannot be displayed in FSCAN.

Database Integrity Considerations

How FSCAN treats the changes you make to the database depends on whether the database is protected by shadow paging.

If you are using shadow paging, FSCAN writes your changes to a shadow database. If you enter the commands END, FILE, or SAVE, the changes become part of the real database. If you enter the command QQUIT or if FSCAN terminates abnormally, the changes disappear and the database is not affected.

If you are not using shadow paging, FSCAN writes your changes directly to the database. The changes remain even after you enter the QQUIT command.

FOCUS performs shadow paging using the Absolute File Integrity facility.

Note: Absolute File Integrity and shadow paging are not supported for XFOCUS data sources.

DBA Considerations

If the database is protected by the DBA security facility, then the ACCESS attribute in the Master File restricts users in the following way:

- Users with read-write access (ACCESS=RW) and write-only access (ACCESS=W) have unrestricted access to the database, with the exception of what is denied them by the RESTRICT and NAME attributes.
- Users with update-only access (ACCESS=U) can display the entire database, with the exception of what is denied them by the RESTRICT and NAME attributes. However, they cannot input or delete instances and can only update non-key fields.
- Users with read-only access (ACCESS=R) to any part of the database cannot use FSCAN on the database.

FSCAN honors DBA security restrictions on segments and fields. FSCAN does not display those segments and fields from which the user is restricted. FSCAN does not honor DBA field value restrictions and will display all field values regardless of the user.

If the user has no access to a key field in the root segment, that user is blocked from using FSCAN on the database.

If the user has no access to a segment, that segment is not listed on the menu that appears when the user enters the CHILD command.

Entering FSCAN

Enter the full-screen FSCAN facility from FOCUS with

```
FSCAN FILE filename
```

where:

filename

Is the name of the database you are editing. The database must be a FOCUS database. You may also enter FSCAN by typing:

```
FS FILE filename
```

For example, to edit the EMPLOYEE database, enter:

```
FSCAN FILE EMPLOYEE
```

Entering FSCAN With a SHOW List

By default, FSCAN makes all fields in the database available to the user. However, it is possible to restrict the fields available with the SHOW option.

Enter FSCAN With a SHOW List

```
FSCAN FILE filename SHOW
[fieldname......fieldname....|SEG.fieldname]
END
```

where:

SHOW

Indicates that specific fields will be displayed. The SHOW keyword must appear on the same line as the FSCAN command.

fieldname...

Are the fields to be displayed.

END

Is required and must be specified on a line by itself.

Entering FSCAN With a SHOW List

For example, the commands

```
FSCAN FILE EMPLOYEE SHOW
EMP_ID LAST_NAME FIRST_NAME SEG.GROSS
END
```

would provide access to only the selected fields in the root segment and to the whole segment containing the field GROSS. The above commands would produce the following display:

```
FSCAN   FILE   EMPLOYEEFOCUS   A               CHANGES :0

         EMP_ID       LAST_NAME    FIRST_NAME
------   ---------    ---------    ----------
==       071382660    STEVENS      ALFRED
==       112847612    SMITH        MARY
==       117593129    JONES        DIANE
==       119265415    SMITH        RICHARD
==       119329144    BANNING      JOHN
==       123764317    IRVING       JOAN
==       126724188    ROMANS       ANTHONY
==       219984371    MCCOY        JOHN
==       326179357    BLACKWOOD    ROSEMARIE
==       451123478    MCKNIGHT     ROGER
==       543729165    GREEENSPAN   MARY
----------------------------INPUT-------------------------------------
==
==>

                                                                MORE=>
```

The only child segment that can be displayed is the SALINFO segment, which contains the field GROSS.

Allowing Uppercase and Lowercase Alpha Fields

By default, FSCAN translates all input and changed alpha fields to uppercase. If uppercase and lowercase input and updates are to be respected, then enter FSCAN with the LOWER keyword.

Specify Case Sensitivity in FSCAN

```
FSCAN FILE filename [case]
```

where:

case

Is one of the following:

UPPER translates all input and changed alpha fields into uppercase. UPPER is the default.

LOWER preserves uppercase and lowercase input and is analogous to the CRTFORM LOWER statement in MODIFY.

MIXED is a synonym for LOWER.

Using FSCAN

When you enter FSCAN, FSCAN displays as much as it can of the root segment of the data source. For example, if you view the EMPLOYEE data source with FSCAN, using the following command

```
FSCAN FILE EMPLOYEE
```

you will see the following screen:

```
1. FSCAN FILE   EMPLOYEEFOCUS   A1              CHANGES:     0

2. EMP_ID        LAST_NAME     FIRST_NAME     HIRE_DATE      DEPARTMENT
   ------        ---------     ----------    ---------  ----------
3. == 071382660  STEVENS       ALFRED         800602         PRODUCTION
4. == 112847612  SMITH         MARY           810701         MIS
   == 117593129  JONES         DIANE          820501         MIS
   == 119265415  SMITH         RICHARD        820104         PRODUCTION
   == 119329144  BANNING       JOHN           820801         PRODUCTION
   == 123764317  IRVING        JOAN           820104         PRODUCTION
   == 126724188  ROMANS        ANTHONY        820701         PRODUCTION
   == 219984371  MCCOY         JOHN           810701         MIS
   == 326179357  BLACKWOOD     ROSEMARIE      820401         MIS
```

```
   == 451123478  MCKNIGHT      ROGER          820202         PRODUCTION
   ------------------------------INPUT-----------------------------------
5. ==
6. ==>
7.                                                                MORE=>
```

This screen displays the contents of the root segment of the EMPLOYEE database. Each record on the screen is one instance in the root segment. The numbers in the diagram refer to the notes below:

1. The header shows the name of the database and the number of changes made to the database since the last save.
2. Each field is labeled with a column heading.
3. The first record at the top of the screen is called the current instance. Many commands operate only on this record. When you first enter FSCAN, this record is the first instance in the root segment.
4. The equal signs (==) in the left margin of the screen indicate the prefix area. This is where you enter prefix area commands.

   The key field value in each record appears highlighted.
5. The last line with equal signs is called the input area and is reserved exclusively for input.
6. The arrow at the lower-left corner of the screen points to the command line. This is where you enter FSCAN commands.
7. The MORE symbol at the lower-right corner of the screen indicates that each record extends to the right of the screen.

This section discusses various functions of the FSCAN facility. For an alphabetic summary of commands, see Syntax Summary.

The FSCAN facility displays one segment at one time. (For the root segment, FSCAN displays all instances in the segment; for descendant segments, FSCAN displays all instances sharing the same parent instance.) Each record on the screen is one segment instance. The first instance at the top of the screen is called the current instance.

The FSCAN facility also displays segments in SINGLE mode, that is, one instance at one time. SINGLE mode is discussed in Displaying a Single Instance on One Screen: The SINGLE and MULTIPLE Commands.

Note the different types of commands:

- **Prefix area commands** are typed in the prefix area on the left of the screen display. Prefix area commands operate only on the line where they are typed.
- **Command-line commands** are typed on the command line at the bottom of the screen. Some commands operate on the entire screen, others operate only on the current instance at the top of the screen. There are two types of command-line commands:
    - **Immediate commands.** When you execute an immediate command, the database remains unchanged even if you typed changes on the screen. There are five immediate commands:

      ```
      LEFT
      RIGHT
      RESET
      ?
      QQUIT
      ```

    - **Non-immediate commands.** When you execute a non-immediate command, any changes you type on the screen will be written to the database even if the command itself does not modify the database.

The following rules apply to commands:

- You may use unique truncations for commands. When this section specifies a command syntax, the unique truncation is shown in uppercase.
- Commands that use field names as parameters require the full field name, alias, or unique truncation.
- You may enter two commands at one time by separating the commands with a semicolon. For example, to enter the commands NEXT 5 and CHILD at one time, type:

  ```
  NEXT 5; CHILD
  ```

The FSCAN Facility and ibi FOCUS Structures

This section is a brief summary of FOCUS structures and how they affect the FSCAN facility.

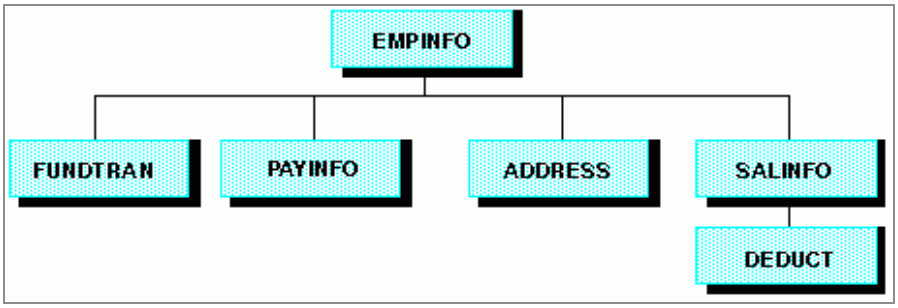
FOCUS databases are organized into segments which have the following properties:

- Segments consist of individual data records called segment instances, in which fields

have a one-to-one correspondence with each other.

- Segments relate to each other as parents and children.
- A group of instances in a child segment describes one instance in a parent segment.
- One parent segment may have many child segments, but a child segment may have only one parent.
- A FOCUS structure has one segment from which all other segments are descended. This is called the root segment.

The diagram below represents the structure of the EMPLOYEE database:



Note the position of the segments in the structure:

- The EMPINFO segment is the root segment. All other segments are descended from it.
- EMPINFO has four children: the FUNDTRAN, PAYINFO, ADDRESS, and SALINFO segments.
- The SALINFO segment has one child, the DEDUCT segment.

The FSCAN facility displays instances in one segment at one time. When it displays the root segment (as it will when you first enter FSCAN), it displays all the instances in the segment.

The following screen illustrates how FSCAN displays the EMPINFO segment.

```
FSCAN   FILE   EMPLOYEEFOCUS    A1             CHANGES :0

        EMP_ID      LAST_NAME  FIRST_NAME  HIRE_DATE   DEPARTMENT
        ------      ---------  ---------   --------    ----------
==      071382660   STEVENS    ALFRED      800602      PRODUCTION
==      112847612   SMITH      MARY        810701      MIS
==      117593129   JONES      DIANE       820501      MIS
```

```
==      119265415   SMITH      RICHARD     820104      PRODUCTION
==      119329144   BANNING    JOHN        820801      PRODUCTION
==      123764317   IRVING     JOAN        820104      PRODUCTION
==      126724188   ROMANS     ANTHONY     820701      PRODUCTION
==      219984371   MCCOY      JOHN        810701      MIS
==      326179357   BLACKWOOD  ROSEMARIE   820401      MIS
==      451123478   MCKNIGHT   ROGER       820202      PRODUCTION
---------------------------------INPUT---------------------------------
--
==
==>
                                                            MORE=>
```

Note that the screen only displays the first five fields of the first ten instances in the segment. To view the other fields and instances, use the scrolling facilities described in Scrolling the Screen.

Also note that you cannot move from one segment to another by simply scrolling. To move from a parent segment to a child segment and back again, you must use the PARENT and CHILD commands discussed in Displaying Descendant Segments: The CHILD, PARENT, and JUMP Commands.

When FSCAN displays a child segment, it displays only those instances relating to an instance in the parent segment. You can scroll back and forth to view all the instances in the group, but you cannot scroll to view the child instances of another parent. At the top of the screen, FSCAN displays up to five keys of the parent instance, and of the parent of the parent, and so on, up to the root segment.

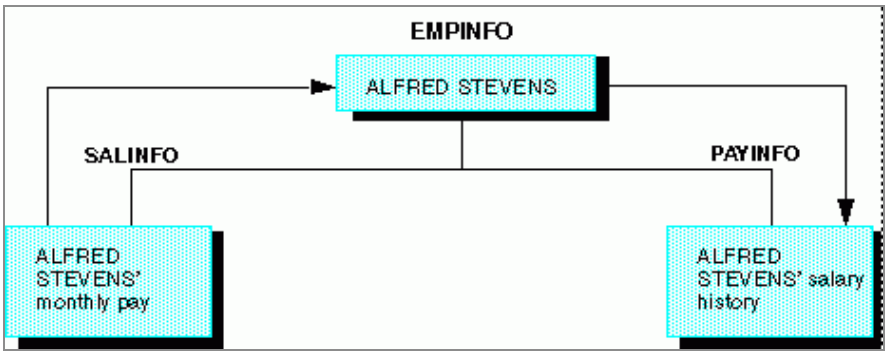
For example, the EMPINFO segment contains the ID numbers and names of employees; its child (SALINFO) contains monthly pay instances. (Each instance lists how much each employee was paid each month.) Each group of instances in SALINFO represents all the monthly pay of one employee recorded in the EMPINFO segment. When FSCAN displays the SALINFO segment, it displays one group of instances at one time.

This is how FSCAN displays the monthly pay of Alfred Stevens, who is listed in the EMPINFO segment. Note that Mr. Stevens' employee ID (the EMPINFO key field) appears at the top of the screen:

```
FSCAN   FILE   EMPLOYEEFOCUS    A1               CHANGES :0

EMP_ID : 071382660

          PAY_DATE     GROSS
          --------     -----
==        820831       916.67
==        820730       916.67
==        820630       916.67
==        820528       916.67
==        820430       916.67
==        820331       916.67
==        820226       916.67
==        820129       916.67
==        811231       833.33
---------------------------------INPUT-----------------------------------
--
==


==>
```

If you are displaying one child segment and wish to display another one, you must return to the parent and request the other child segment. For example, if you are examining Alfred Stevens' monthly pay and wish to view his salary history (contained in the segment PAYINFO), return to the EMPINFO segment and request PAYINFO information for Alfred Stevens using the CHILD command described in Displaying Descendant Segments: The CHILD, PARENT, and JUMP Commands.

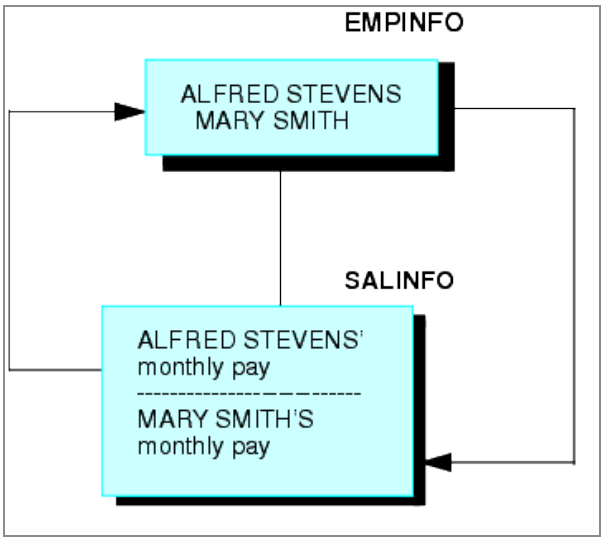
The figure below shows this path schematically. The arrows show the direction you are traveling to move from the SALINFO segment to the PAYINFO segment:



Similarly, if you are displaying one group of child instances and wish to display another group within the same segment but belonging to a different instance in the parent, you must return to the parent segment and request the child segment for the other instance.

For example, suppose you are examining Alfred Stevens' monthly pay and wish to view Mary Smith's monthly pay. You must return to the EMPINFO segment and select the SALINFO segment for Mary Smith.

The figure below shows this path schematically. The arrows show the direction you are traveling to move from Alfred Stevens' monthly pay instances to Mary Smith's monthly pay instances:



Scrolling the Screen

You may scroll the screen forward and backward, right and left.

Scroll the Screen Forward

To scroll forward one screen in a segment, enter

```
FOrward
```

or press the PF8 or PF20 key. Note that the last instance on one screen becomes the first instance on the next screen.

To scroll the screen *n* lines forward, enter

```
Next n
```

or:

```
DOwn n
```

If you do not enter a number for *n*, the default is 1.

Scrolling Forward

For example, suppose the screen displays the EMPLOYEE root segment as shown below.

```
FSCAN FILE EMPLOYEEFOCUS A1         CHANGES :0

     EMP_ID      LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
     ------      ---------  ----------  ---------  ----------
==   071382660   STEVENS    ALFRED      800602     PRODUCTION
==   112847612   SMITH      MARY        810701     MIS
==   117593129   JONES      DIANE       820501     MIS
==   119265415   SMITH      RICHARD     820104     PRODUCTION
==   119329144   BANNING    JOHN        820801     PRODUCTION
==   123764317   IRVING     JOAN        820104     PRODUCTION
==   126724188   ROMANS     ANTHONY     820701     PRODUCTION
==   219984371   MCCOY      JOHN        810701     MIS
==   326179357   BLACKWOOD  ROSEMARIE   820401     MIS
==   451123478   MCKNIGHT   ROGER       820202     PRODUCTION
---------------------------------INPUT---------------------------------
==
==> forward

                                                              MORE=>
```

When you type the FORWARD command on the command line and press **Enter**, the following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1           CHANGES :0

     EMP_ID      LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
     ------      ---------  ----------  ---------  ----------
==   451123478   MCKNIGHT   ROGER       820202     PRODUCTION
==   543729165   GREENSPAN  MARY        820401     MIS
==   818692173   CROSS      BARBARA     811102     MIS









----------------------------------INPUT---------------------------------
==
==>

                                                                MORE=>
```

Scroll the Screen Backward

To scroll the screen backward, enter

```
Backward
```

or press the PF7 or PF19 key.

Scroll the Screen to the Right and the Left

To scroll the screen one panel to the right, enter

```
RIght
LEft
```

or press the PF11 or PF23 key.

To scroll the screen one panel to the left, enter

```
LEft
```

or press the PF10 or PF22 key.

The commands RIGHT and LEFT are immediate commands. When you scroll right and left, FSCAN does not enter changes you typed on the screen until you press Enter after scrolling.

Scrolling the Screen

For example, if you scroll the EMPLOYEE root segment display one panel to the right, the following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1         CHANGES :0

    CURR_SAL         CURR_JOBCODE    ED_HRS
    --------         ------------    ------
==  11000.00         A07              25.00
==  13200.00         B14              36.00
==  18480.00         B03              50.00
==  9500.00          A01              10.00
==  29700.00         A17                .00
==  26862.00         A15              30.00
==  21120.00         B04               5.00
==  18480.00         B02                .00
==  21780.00         B04              75.00
==  16100.00         B02              50.00
---------------------------------INPUT--------------------------------
==
==>

                                                                MORE=>
```

Selecting a Specific Instance by Defining a Current Instance

This section describes how to move through the database by defining a particular instance as the current instance. The current instance is always the top instance on the screen. Certain commands only operate on the current instance.

Define a Current Instance

To define an instance as the current instance, type a slash (/) in the prefix area corresponding to the instance.

You may also type a slash before or after the following prefix area commands:

- The K command (K/ or /K). After FSCAN changes the key field and displays the instance in proper sequence, it makes the instance the current instance.
- The I command (I/ or /I). After FSCAN adds a new instance to the database, it makes the instance the current instance.

Defining a Current Instance: The "/" Prefix

For example, suppose you type a slash in the prefix area of John Banning's instance, as shown below:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0

   EMP_ID     LAST_NAME   FIRST_NAME  HIRE_DATE   DEPARTMENT
   ------     ---------   ----------  ----------  ---------
== 071382660  STEVENS     ALFRED      800602      PRODUCTION
== 112847612  SMITH       MARY        810701      MIS
== 117593129  JONES       DIANE       820501      MIS
== 119265415  SMITH       RICHARD     820104      PRODUCTION
/= 119329144  BANNING     JOHN        820801      PRODUCTION
== 123764317  IRVING      JOAN        820104      PRODUCTION
```

```
== 126724188  ROMANS      ANTHONY     820701      PRODUCTION
== 219984371  MCCOY       JOHN        810701      MIS
== 326179357  BLACKWOOD   ROSEMARIE   820401      MIS
== 451123478  MCKNIGHT    ROGER       820202      PRODUCTION
---------------------------------INPUT-------------------------------
==
==>
                                                              MORE=>
```

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

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0

   EMP_ID     LAST_NAME   FIRST_NAME  HIRE_DATE   DEPARTMENT
   ------     ---------   ----------  ---------   ----------
== 119329144  BANNING     JOHN        820801      PRODUCTION
== 123764317  IRVING      JOAN        820104      PRODUCTION
== 126724188  ROMANS      ANTHONY     820701      PRODUCTION
== 219984371  MCCOY       JOHN        810701      MIS
== 326179357  BLACKWOOD   ROSEMARIE   820401      MIS
== 451123478  MCKNIGHT    ROGER       820202      PRODUCTION
== 543729165  GREENSPAN   MARY        820401      MIS
== 818692173  CROSS       BARBARA     811102      MIS
---------------------------------INPUT-------------------------------
==
==>
                                                              MORE=
```

Define the First and Last Instances of a Segment on Display: The FIRST, LAST, and TOP Commands

FSCAN displays all instances in a segment that share a common parent instance. For the root segment, this means all the instances in the segment. To define the first instance in the group as the current instance, enter:

```
FIrst
```

If you are displaying instances in the root segment, FIRST will make the first instance in the database the current instance. If you are displaying instances in a child segment and use the FIRST command, the first child instance will become the current instance.

To define the last instance as the current instance, enter:

```
LAst
```

To select the first instance in the root segment of the database to be the current instance, enter:

```
Top
```

TOP displays the root segment, scrolled to the leftmost panel, with the first instance the current instance.

Defining the Last Instance as the Current Instance With LAST

For example, if you enter LAST on the EMPLOYEE root segment display, the following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0

   EMP_ID    LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
   ------    ---------  ----------  ---------  ----------
== 818692173 CROSS      BARBARA     811102     MIS
```

```
--------------------------------INPUT-----------------------------------
==
==>

                                                                MORE=>
```

Locate an Instance Based on Field Values: The LOCATE Command

LOCATE searches for instances containing field values that fulfill certain conditions. For example, it can search for an instance with a LAST_NAME value of BANNING or a CURR_SAL value less than 20,000. LOCATE searches starting with the current instance.

The syntax is (entered on one line)

```
LOcate field1 rel1 value1 [OR value1a OR value1b OR ...]

[{AND|,} field2 rel2 value2 {AND|,} ...]
```

where:

fieldn ...

Is a field to be tested.

reln ...

Is one of the following condition relations:

`EQ` or `=`	Equal to
`NE`	Not equal to

`GE`	Greater than or equal to
`GT`	Greater than
`LE`	Less than or equal to
`LT`	Less than
`CONTAINS` or `CO`	Contains the character string
`OMITS` or `OM`	Omits the character string

valuen ...

Is a value for which FSCAN can test. The first instance with a field value that passes the test becomes the current segment.

If you supply more than one test condition in the command, FSCAN searches for the instance that fulfills all of the conditions. Separate the test conditions in the command with the word AND or with a comma (,).

OR

Enables you to test a field for multiple values. If the field contains one of the values, it meets the test. You can use AND and OR in a single LOCATE command.

The LOCATE command searches starting with the first instance following the current instance. If LOCATE cannot find the instance, it displays a message and the current instance does not change.

Locating an Instance Based on Field Values

For example, suppose the first instance in the EMPLOYEE root segment is the current instance. If you issue the command

```
LOCATE LAST_NAME EQ SMITH
```

the following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1        CHANGES :0

   EMP_ID      LAST_NAME      FIRST_NAME     HIRE_DATE      DEPARTMENT
   ------      ---------      ----------     ---------       ----------
== 112847612   SMITH          MARY           810701         MIS
== 117593129   JONES          DIANE          820501         MIS
== 119265415   SMITH          RICHARD        820104         PRODUCTION
== 119329144   BANNING        JOHN           820801         PRODUCTION
== 123764317   IRVING         JOAN           820104         PRODUCTION
== 126724188   ROMANS         ANTHONY        820701         PRODUCTION
== 219984371   MCCOY          JOHN           810701         MIS
== 326179357   BLACKWOOD      ROSEMARIE      820401         MIS
== 451123478   MCKNIGHT       ROGER          820202         PRODUCTION
== 543729165   GREENSPAN      MARY           820401         MIS
--------------------------------INPUT-----------------------------------
==
==>
                                                                MORE=>
```

These are other examples of the LOCATE command:

```
LOCATE JOBCODE EQ A07 OR A17
```

This LOCATE searches for the first segment instance that has a JOBCODE value of either A07 or A17.

```
LOCATE LAST_NAME CO WOOD
```

This LOCATE searches for the first segment instance with a LAST_NAME value that contains the character string WOOD.

```
LOCATE HIRE_DATE GT 820401 AND JOBCODE IS B02 OR B03
```

This LOCATE searches for the first segment instance with both a HIRE_DATE value greater than 820401 and a JOBCODE value that is either B02 or B03.

Find an Instance in a Group: The FIND Command

The FIND command works within the group of instances being displayed. In the root segment, this is all instances in the segment; in descendant segments, this is all instances sharing a common parent instance. FIND searches for instances containing field values that fulfill certain conditions. For example, it can search for an instance with a LAST_NAME value of BANNING or a CURR_SAL value less than 20,000. FIND searches starting with the current instance.

The syntax is entered on one line.

```
FIND field1 rel1 value1 [OR value1a OR value1b OR ...]
[{AND|,} field2 rel2 value2 {AND|,} ...]
```

```
[{AND|,} field2 rel2 value2 {AND|,} ...]
```

where:

fieldn ...

Is a field in the segment.

reln ...

Is one of the following condition relations:

`EQ` or `=`	Equal to
`NE`	Not equal to

`GE`	Greater than or equal to
`GT`	Greater than
`LE`	Less than or equal to
`LT`	Less than
`CONTAINS` or `CO`	Contains the character string
`OMITS` or `OM`	Omits the character string

valuen ...

Is a value for which FSCAN can test. The first instance with a field value that passes the test becomes the current segment.

If you supply more than one test condition in the command, FSCAN searches for the instance that fulfills all of the conditions. Separate the test conditions in the command with the word AND or with a comma (,).

OR

Enables you to test a field for multiple values. If the field contains one of the values, it meets the test. You can use AND and OR in a single FIND command.

The FIND command searches the group starting with the first instance following the current instance. To search the entire group, issue the FIRST command before issuing FIND. If FIND cannot find the instance, it displays a message and the current instance does not change.

Finding an Instance in a Group

For example, suppose the first instance in the EMPLOYEE root segment is the current instance. If you issue the command

```
FIND LAST_NAME EQ SMITH
```

the following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0

    EMP_ID     LAST_NAME     FIRST_NAME  HIRE_DATE  DEPARTMENT
    ------     ---------     ----------  ---------  ----------
==  112847612  SMITH         MARY        810701     MIS
==  117593129  JONES         DIANE       820501     MIS
==  119265415  SMITH         RICHARD     820104     PRODUCTION
==  119329144  BANNING       JOHN        820801     PRODUCTION
==  123764317  IRVING        JOAN        820104     PRODUCTION
==  126724188  ROMANS        ANTHONY     820701     PRODUCTION
==  219984371  MCCOY         JOHN        810701     MIS
==  326179357  BLACKWOOD     ROSEMARIE   820401     MIS
==  451123478  MCKNIGHT      ROGER       820202     PRODUCTION
==  543729165  GREENSPAN     MARY        820401     MIS
---------------------------------INPUT--------------------------------
==
==>

                                                            MORE=>
```

These are other examples of the FIND command:

```
FIND DEPARTMENT EQ MIS OR SALES
```

This FIND searches for the first segment instance that has a DEPARTMENT value of either MIS or SALES.

```
FIND LAST_NAME CO WOOD
```

This FIND searches for the first segment instance with a LAST_NAME value that contains the character string WOOD.

```
FIND HIRE_DATE GT 820401 AND DEPARTMENT EQ MIS OR PRODUCTION
```

This FIND searches for the first segment instance with both a HIRE_DATE value greater than 820401 and a DEPARTMENT value that is either MIS or PRODUCTION.

Displaying Descendant Segments: The CHILD, PARENT, and JUMP Commands

The CHILD, PARENT, and JUMP commands enable you to display the data in different segments of a data source.

Display a Child Segment

To display instances in a child segment relating to the current instance, enter

```
CHIld
```

or press PF5 or PF17. If the segment on the screen when you enter the command has only one child segment, FSCAN shows the child segment. If the segment on the screen has more than one child segment, FSCAN displays a menu of child segments. Select a segment by entering its number. (**Note:** The menu does not display segments restricted to you as a result of DBA restrictions.)

If you already know the number of the segment on the menu, you can skip the menu by entering

```
CHIld n
```

where:

n

is the number of the segment on the menu.

You can display the child instances of any instance on the screen by typing C in the prefix area next to the instance. You can skip the menu by typing C followed by the number of the segment on the menu.

Displaying a Child Segment

For example, suppose you are displaying the root segment of the EMPLOYEE database and you want to see the monthly pay of Mary Smith. Monthly pay is contained in the segment SALINFO, a child of the root segment. First, make Mary Smith's instance the current instance. Then, enter the command:

```
CHILD
```

The following menu appears:

```
FSCAN FILE EMPLOYEEFOCUS A1              CHANGES :0


 Please enter the number of the child segment you want


 1)FUNDTRAN            2)PAYINFO
 3)ADDRESS             4)SALINFO









 =>
    Enter the number of the child you want
    Enter 0 to stay at parent.
```

Enter the number 4. The following screen appears:

```
 FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0

 EMP_ID : 112847612

          PAY_DATE      GROSS
          --------      -----

 ==       820831        1100.00
 ==       820730        1100.00
 ==       820630        1100.00
 ==       820528        1100.00
 ==       820430        1100.00
 ==       820331        1100.00
 ==       820226        1100.00
 ==       820129        1100.00

--------------------------------INPUT-----------------------------------
 ==
 ==>
```

Note that the header displays the key field value of the parent instance. Since EMP_ID is the key field of the root segment, the header displays Mary Smith's employee ID.

Also, you could have gone directly from the EMPLOYEE root segment to the monthly pay segment by doing one of the following:

- Typing CHILD 4 on the command line.
- Typing C4 in the prefix area.

Display the Parent Segment

To return to the parent segment, enter

```
Parent
```

or press PF4 or PF16. The current instance in the parent is the same as before you entered the CHILD command or C prefix area command.

Display the First Child of the Next Parent Instance

To move to the first child of the next parent instance, enter

```
JUMP
```

or press PF12 or PF24 while FSCAN is displaying a child segment.

Displaying the First Child of the Next Parent Instance

For example, if you enter JUMP while the PAYINFO segment is being displayed for a particular employee, the PAYINFO segment for the next employee in the EMP_INFO segment is displayed. JUMP may be issued anywhere.

Displaying a Single Instance on One Screen: The SINGLE and MULTIPLE Commands

To display a single instance on the screen, enter:

```
SIngle
```

This places you in SINGLE mode. SINGLE mode enables you to view a single segment instance on one screen. Only the current instance appears, but all its fields appear on one screen (unless it has many fields). You may enter all FSCAN commands on the command line at the bottom of the screen, but there is no prefix area. The key field values appear highlighted.

All FSCAN commands (but not prefix area commands) operate in SINGLE mode, except that only one instance is displayed. In particular, note the following:

- If you enter the FORWARD command in SINGLE mode, FSCAN displays the next

instance in the segment. If you enter the BACKWARD command, FSCAN displays the previous instance.

- If you enter the CHILD command, only one child instance appears at one time. If you enter the PARENT command, only the parent instance of the current instance appears on the screen.

You can update and delete an instance in SINGLE mode, but you cannot add another instance.

You remain in SINGLE mode until you enter the command:

```
Multiple
```

MULTIPLE returns you to normal mode, which displays multiple instances at one time.

Using SINGLE Mode

For example, this is how Diane Jones' instance looks in SINGLE mode. Note that there is no input area, and that the arrow at the bottom of the screen points to the command line where you can enter commands:

```
  FSCAN FILE EMPLOYEEFOCUS A1                CHANGES : 0

       EMP_ID : 117593129     LAST_NAME : JONES
   FIRST_NAME : DIANE         HIRE_DATE : 820501
   DEPARTMENT : MIS            CURR_SAL : 18480.00
 CURR_JOBCODE : B03               ED_HRS : 50.00




  ==>
```

Modifying the Database

You may use FSCAN to modify the database by adding, updating, and deleting segment instances.

Adding New Segment Instances: The "I" Prefix

To add a new segment instance to the segment displayed on the screen, type the instance field values in the input area on the bottom of the screen. You can use the Tab key to jump from field to field. Then type I in the prefix area next to the new instance. When you press Enter, FSCAN adds the instance to the database, displaying it in proper sequence based on its key field values.

If the instance you are typing extends beyond the right margin of the screen, use the scrolling commands discussed in Scrolling the Screen. FSCAN adds the segment instance when you press Enter or enter any command except RIGHT, LEFT, RESET, ?, and QQUIT.

Note:

- FSCAN does not accept new instances with key field values that are the same as another instance.
- FSCAN does not accept new instances with field values that do not conform to the ACCEPT attribute in the Master File (ACCEPT is explained in the *Describing Data* manual).
- If you want the new instance to become the current instance, type I/ in the prefix area next to the new instance before pressing Enter.

Adding New Segment Instances

For example, suppose you want to add Fred Johnson to the EMPLOYEE database, and you want the new instance to become the current instance. Type his instance in the input area as shown below (note the I/ in the prefix area):

```
FSCAN FILE EMPLOYEEFOCUS A1     CHANGES :0
     EMP_ID     LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
     ------     ---------  ----------  ---------  ----------
==   117593129  JONES      DIANE       820501     MIS
==   119265415  SMITH      RICHARD     820104     PRODUCTION
==   119329144  BANNING    JOHN        820801     PRODUCTION
==   123764317  IRVING     JOAN        820104     PRODUCTION
==   126724188  ROMANS     ANTHONY     820701     PRODUCTION
```

```
==   219984371  MCCOY      JOHN        810701     MIS
==   326179357  BLACKWOOD  ROSEMARIE   820401     MIS
==   451123478  MCKNIGHT   ROGER       820202     PRODUCTION
==   543729165  GREENSPAN  MARY        820401     MIS
==   818692173  CROSS      BARBARA     811102     MIS
----------------------------INPUT-------------------------------------
I/   123123123  johnson    fred        870507     mis

==>
                                                              MORE=>
```

When you press Enter, the screen appears as follows:

```
FSCAN FILE EMPLOYEEFOCUS A1          CHANGES :1

     EMP_ID     LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
     ------     ---------  ---------   ---------  ----------
==   123123123  JOHNSON    FRED        870507      MIS
==   123764317  IRVING     JOAN        820104      PRODUCTION
==   126724188  ROMANS     ANTHONY     820701      PRODUCTION
==   219984371  MCCOY      JOHN        810701      MIS
==   326179357  BLACKWOOD  ROSEMARIE   820401      MIS
==   451123478  MCKNIGHT   ROGER       820202      PRODUCTION
==   543729165  GREENSPAN  MARY        820401      MIS
==   818692173  CROSS      BARBARA     811102      MIS
----------------------------INPUT-------------------------------------
==

==>
  0 Keys Changed 0 Non-Keys Changed
  0 Records Deleted 1 Records Input

                                                              MORE=>
```

If you do not type "I" in the prefix area when you input a new instance, FSCAN displays an error message. To continue, you must do one of the following:

- Enter "I" in the prefix area of the input area.
- Cancel the input by entering the RESET command, typing R in the prefix area, or pressing the PF2 or PF14 key. This also recovers typed-over field values (see the following section).

Note that the RESET command entered on the command line is an immediate command. However, the R prefix-area command is not an immediate command. If you typed changes on a line not specifying the R prefix, FSCAN enters the changes.

Updating Non-Key Field Values

There are three ways to update non-key field values:

- Type over field values.
- Issue the REPLACE command.
- Issue the CHANGE command.

Note that FSCAN does not accept any new field value that does not conform to the ACCEPT attribute in the Master File (the ACCEPT attribute is explained in the *Describing Data* manual).

Type Over Field Values

You may update segment instances by typing over their values on the screen. Use the Tab key to jump from field to field within the same instance.

Typing Over Field Values

For example, suppose you want to change Richard Smith's department from Production to Sales. Simply type over the DEPARTMENT value and press **Enter**. The screen appears as shown on the next page. Note that the message at the bottom of the screen indicates one changed non-key field.

The screen is:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :0
```

```
      EMP_ID     LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
      ------     ---------  ----------  ---------  ----------
==    071382660  STEVENS    ALFRED      800602     PRODUCTION
==    112847612  SMITH      MARY        810701     MIS
==    117593129  JONES      DIANE       820501     MIS
==    119265415  SMITH      RICHARD     820104     SALES
==    119329144  BANNING    JOHN        820801     PRODUCTION
==    123764317  IRVING     JOAN        820104     PRODUCTION
==    126724188  ROMANS     ANTHONY     820701     PRODUCTION
==    219984371  MCCOY      JOHN        810701     MIS
==    326179357  BLACKWOOD  ROSEMARIE   820401     MIS
==    451123478  MCKNIGHT   ROGER       820202     PRODUCTION
-------------------------------INPUT---------------------------------
==
==>
  0 Keys Changed 1 Non-Keys Changed
  0 Records Deleted 0 Records Input

                                                              MORE=>
```

The message at the bottom of the screen indicates the number of field values you changed since the last time you pressed Enter. The counter at the top of the screen counts the total number of values you changed since the last time the changes were saved on disk.

If you type over field values and change your mind before you press Enter, you can restore the original field values by entering R (to specify the RESET command) on the prefix area next to the instance whose values you are recovering, or by pressing the PF2 or PF14 key. However, if you press Enter before pressing one of these keys, you will not recover the typed-over values.

Note that the RESET command entered on the command line is an immediate command. However, the R prefix area command is not an immediate command. If you typed changes on a line not specifying the R prefix, FSCAN enters the changes.

Replace Field Values: The REPLACE Command

The REPLACE command replaces one field value with another either for a specific instance or for all the instances in a group. (In the root segment, this is all the instances in the segment; in a descendant segment, this is all the instances that share a parent instance.) The syntax is

```
REPlace field1 = value1[,field2 = value2, ...] [,$ {*|n}]
```

where:

fieldn ...

Is a field in the current instance whose value you want to change.

valuen ...

Is a new value for the field.

,$ {*|n}

Enables you to change multiple instances starting from the current instance (the current instance included). *n* is the number of instances to be searched for the field value you want to change. If you want all instances in the group starting from the current instance changed, use an asterisk (*).

Using REPLACE

For example, to change Richard Smith's department from Production to Sales, make Richard Smith's instance the current instance. Then enter:

```
REPLACE DEPARTMENT = SALES
```

To change the DEPARTMENT value to SALES in the next five instances, enter:

```
REPLACE DEPARTMENT = SALES,$ 5
```

To change all DEPARTMENT values in the group to SALES, make the first instance on display the current instance by entering:

```
FIRST
```

Then enter:

```
REPLACE DEPARTMENT = SALES,$ *
```

Change Character Strings Within Field Values: The CHANGE Command

The CHANGE command changes character strings within field values either for a specific instance or for all the instances in a group (in the root segment, this is all the instances in the segment; in a descendant segment, this is all the instances that share a parent instance). The fields must be alphanumeric. The syntax is

```
CHAnge field = /oldstring/newstring/ [,$ {*|n}]
```

where:

field

Is the name of the field in the current instance whose value you want to change. The field must be alphanumeric, and it cannot be a key field.

oldstring

Is the substring of the field value that you want to change.

newstring

Is the character string to replace the substring.

,$ {*|n}

Enables you to change multiple instances counting from the current instance (the current instance included). *n* is the number of instances to be searched for the substring. If you want all instances in the group searched, starting from the current instance, use an asterisk (*).

Using CHANGE

For example, to change Joan Irving's department from Production to Products, make Joan Irving's instance the current instance. Then enter:

```
CHANGE DEPARTMENT = /ION/S/
```

To change the Production department to Products in the next five instances starting from the current instance, enter:

```
CHANGE DEPARTMENT = /ION/S/,$ 5
```

To change this substring in all the instances in the group, make the first instance on display the current instance by entering:

```
FIRST
```

Then enter:

```
CHANGE DEPARTMENT = /ION/S/ ,$ *
```

Changing Key Field Values

FSCAN enables you to change values of key fields, either by typing over the values or by using the REPLACE KEY command.

Note: FSCAN does not allow you to change a key field to a value that will make the key field values of one instance the same as another instance.

FSCAN does not accept any new key field value that does not conform to the ACCEPT attribute in the Master File (the ACCEPT attribute is explained in the *Describing Data* manual).

Type Over Key Field Values: The KEY Command

To change the value of a key field, do the following:

Procedure

1. Type the new value over the old one.
2. Either type a K in the prefix area next to the instance you are changing, or type the command:

```
Key
```

If you want the instance to be the current instance after its key value is changed, type K/ in the prefix area next to the instance.

3. Press Enter.

Result

After you change the key value, FOCUS moves the instance within the segment so that the key values remain sorted in their proper sequence. The screen shows this immediately.

Note: FOCUS does not physically move instances in the root segment, although the instances appear on the FSCAN screen sorted by their key field values.

If you do not enter the KEY command or type K in the prefix area when you change a key field value, FSCAN displays an error message. Before continuing, you must do one of the following:

- Enter the KEY command, or enter K in the prefix area.
- Retype the original key value.
- Restore the key field value by entering the RESET command, typing R in the prefix area, or pressing the PF2 or PF14 key. Other field values you typed over will also be restored.

Note: The RESET command entered on the command line is an immediate command. However, the R prefix area command is not an immediate command. If you type any changes on a line that does not specify the R prefix, FSCAN enters the changes.

Using KEY

For example, suppose you want to change Alfred Stevens' employee ID from 071382660 to 444555666, and you want his instance to remain the current instance. Type over the employee ID and type K/ in the prefix area.

The screen appears as shown below:

```
FSCAN FILE EMPLOYEEFOCUS A1             CHANGES :2

     EMP_ID    LAST_NAME  FIRST_NAME  HIRE_DATE  DEPARTMENT
     ------    ---------  ----------  ---------  ----------
k/   444555666 STEVENS    ALFRED      800602     PRODUCTION
==   112847612 SMITH      MARY        810701     MIS
==   117593129 JONES      DIANE       820501     MIS
==   119265415 SMITH      RICHARD     820104     PRODUCTION
==   119329144 BANNING    JOHN        820801     PRODUCTION
==   123764317 IRVING     JOAN        820104     PRODUCTION
==   126724188 ROMANS     ANTHONY     820701     PRODUCTION
==   219984371 MCCOY      JOHN        810701     MIS
==   326179357 BLACKWOOD  ROSEMARIE   820401     MIS
==   451123478 MCKNIGHT   ROGER       820202     PRODUCTION
-------------------------------INPUT-----------------------------------
==
==>
                                                              MORE=>
```

When you press **Enter**, the screen appears as shown below:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :3

     EMP_ID    LAST_NAME FIRST_NAME HIRE_DATE DEPARTMENT
     ------    --------- ---------- --------- ----------
==   444555666 STEVENS   ALFRED     800602    PRODUCTION
==   451123478 MCKNIGHT  ROGER      820202    PRODUCTION
==   543729165 GREENSPAN MARY       820401    MIS
==   818692173 CROSS     BARBARA    811102    MIS
-------------------------------INPUT-----------------------------------
==
==>
  1 Keys Changed 0 Non-Keys Changed
  0 Records Deleted 0 Records Input

                                                              MORE=>
```

The message at the bottom of the screen indicates the number of key field values you changed since the last time you pressed **Enter**.

Change Key Field Values Using the REPLACE KEY Command

You may also use the REPLACE command to change key fields of the current instance. The syntax of the REPLACE command to replace key fields is

```
REPlace KEY key1 = value1[, key2 = value2, ...]
```

where:

keyn ...

Is the key field you want to change. Remember that an instance may have more than one key field (as determined by the SEGTYPE attribute in the Master File).

valuen ...

Is the new value for the key field.

Using REPLACE KEY

For example, to change Alfred Stevens' employee ID from 444555666 to 071382660, make his instance the current instance by placing a slash in the prefix area, and enter the following:

```
REPLACE KEY EMP_ID = 071382660
```

Deleting Segment Instances: The DELETE Command

You can easily delete a data instance with the DELETE command.

Delete Segment Instances

To delete the current instance, type a D in the prefix area next to the instance or enter:

```
DElete
```

FSCAN displays the complete segment instance alone on the screen and asks if you really want to delete it. Press Enter to delete the instance, or respond:

N

Do not delete the current instance. (Returns to the previous screen.)

Q

Do not delete the current instance. (If you made no other changes to the database, entering Q leaves FSCAN and returns to the FOCUS prompt. Otherwise, it returns to the previous screen.)

Note: When you delete an instance, you delete all its descendant instances as well.

Using DELETE

For example, suppose you want to delete information about John Banning from the database. First, make John Banning's instance the current instance. Then, enter the DELETE command. The following screen appears:

```
FSCAN FILE EMPLOYEEFOCUS A1            CHANGES :4

            Delete Confirmation Screen

           EMP_ID : 119329144          LAST_NAME : BANNING
       FIRST_NAME : JOHN               HIRE_DATE : 820801
       DEPARTMENT : PRODUCTION          CURR_SAL : 29700.00
     CURR_JOBCODE : A17                    ED_HR :    .00
```

```
==>
      Press ENTER to delete
      Enter N(o) to abort
      Enter Q(uit) to quit session
```

If you press Enter, the screen appears as follows:

```
FSCAN FILE EMPLOYEEFOCUS A1CHANGES :2

     EMP_ID    LAST_NAME FIRST_NAME HIRE_DATE DEPARTMENT
     ------    --------- ---------- --------- ----------
==   123764317 IRVING    JOAN       820104    PRODUCTION
==   126724188 ROMANS    ANTHONY    820701    PRODUCTION
==   219984371 MCCOY     JOHN       810701    MIS
==   326179357 BLACKWOOD ROSEMARIE  820401    MIS
==   451123478 MCKNIGHT  ROGER      820202    PRODUCTION
==   543729165 GREENSPAN MARY       820401    MIS
==   818692173 CROSS     BARBARA    811102    MIS
-------------------------------INPUT-----------------------------------
==
==>

0 Keys Changed 0 Non-Keys Changed
1 Records Deleted 0 Records Input

                                                              MORE=>
```

Repeating a Command: ? and =

Two commands help you enter a command repeatedly:

- The ? command displays the last command you entered.
- The = command executes the last command you entered.

Display Previous Commands: The ? Command

To display the last command you entered, enter

```
?
```

or press **PF6** or **PF18**. This displays the previous command on the command line. You may then execute the command by pressing Enter or remove the command from the command line.

As you enter FSCAN commands on the command line, FSCAN stores them in a stack in memory. If you enter the ? command repeatedly, FSCAN scrolls through the stack, displaying the commands in stack from the most recent to the oldest.

The ? command is an immediate command. The database remains unchanged until you press Enter a second time or enter a non-immediate command. Immediate commands were explained previously at the beginning of Using FSCAN.

Executing the Previous Command: The = Command

The = command executes the last command you entered. Enter

```
=
```

or press **PF9** or **PF21**.

Saving Changes: The SAVE Without Exiting FSCAN Command

To save the changes to the database that you made on FSCAN, enter

```
SAve
```

You remain in FSCAN. The counter at the top of the screen that counts changes in the database is reset to 0.

Exiting FSCAN: The END, FILE, QQUIT, and QUIT Commands

To exit FSCAN and save the changes you made to the database, enter

```
End
```

or:

```
FILe
```

If bad data is encountered upon trying to save your changes, an error message is generated.

To exit FSCAN without saving the changes you made to the database, enter:

```
QQuit
```

Note: QQUIT only suppresses changes made on FSCAN when you are using the Absolute File Integrity facility. Otherwise, FSCAN writes all changes to the database.

If you did not make any changes to the database, you can exit FSCAN by entering

```
QUit
```

or by pressing the **PF3** or **PF15** key.

The FSCAN HELP Facility

FSCAN has a HELP facility. To use HELP, enter the command

```
Help
```

or press the **PF1** or **PF13** key. HELP displays a summary of FSCAN commands and prefix area commands, as shown in the sample screen below:

```
FSCAN FILE CAR     FOCUS A        HELP SCREEN 2 of 3

                         FSCAN COMMANDS
=          - Re-execute the most recent command line.
?          - Retrieve the previous command line.
Backward   - Go backward one screen.
CHAnge     - Change a string within a field:
             CHANGE fieldname=/oldstring/newstring/,$
CHIld      - Display child instances of this segment.
DElete     - Delete a segment instance, and all of its children.
DIsplay    - Display the segment containing the specified fieldname.
End/FILe   - Save all changes and exit FSCAN.
FINd       - Find an instance on this chain which satisfies a test:
             FIND fieldname EQ GT CO... value.
FIRst      - Go to the first instance on this chain.
FOrward    - Go forward one screen.
Jump       - Jump to the children of the next parent.
LAst       - Go to the last instance on this chain
LEft       - Go left one panel.
LOcate     - Same as FIND but search is throughout the database.

Exit HELP: PF03/PF15. Forward: PF08/PF20. Backward: PF07/PF19.
```

You can scroll HELP screens back and forth by pressing the **PF8**or **PF20** key to go forward and the **PF7** or **PF19** key to go backward.

To exit the HELP facility, press the **PF3** or **PF15** key.

Syntax Summary

This section is a summary of the FSCAN commands, PF keys, and prefix area commands. References to other sections are included.

Summary of Commands

FSCAN commands are listed here in alphabetical order. The unique truncation of each command is capitalized.

Backward

Scrolls the display one screen backward.

PF keys: PF7 or PF19.

CHAnge

Changes character strings within field values. The syntax is

```
CHAnge field =/oldstring/newstring/ [,$ {*|n}]
```

where:

field

Is the name of the field whose value you want to change. The field must be alphanumeric and it cannot be a key field.

oldstring

Is the substring of the field value that you want to change.

newstring

Is the character string to replace the substring.

,$ {*|n}

Enables you to change multiple instances counting from the current instance (the current instance included). *n* is the number of instances to be searched for the substring. If you want all instances in the group searched (starting from the current instance), use an asterisk (*).

You can also change field values by typing over them.

CHIld

Displays the child instances relating to the current instance. (In SINGLE mode, displays the first child instance of the current instance.) The syntax is

```
CHIld [n]
```

where:

n

Is the number of the child segment as assigned by FSCAN. If you omit this number, FSCAN displays a menu listing the segments and their numbers. Enter a number to display the segment (Displaying Descendant Segments: The CHILD, PARENT, and JUMP Commands).

Prefix area command: `C[n]`

where:

n

Is the number of the child segment as assigned by FSCAN. If you omit this number, FSCAN displays the menu.

DElete

Deletes the current instance and all descendant instances.

Prefix area command: `D`

DOwn [n]

Scrolls the display *n* lines forward. *n* defaults to 1.

DIsplay Field Name

Displays the segment containing the specified field name.

End

Saves all changes made to the database and exits the FSCAN facility (see Exiting FSCAN: The END, FILE, QQUIT, and QUIT Commands).

FILe

Saves all changes made to the database and exits the FSCAN facility (see Exiting FSCAN: The END, FILE, QQUIT, and QUIT Commands).

FINd

Searches a group of instances (in the root segment, this is all instances in the segment; in descendant segments, this is all instances sharing a common parent instance) for an instance containing field values that fulfill certain conditions. FIND searches the group starting from the current instance. If it finds the instance, it makes that instance the current instance.

The syntax is (entered on one line)

```
FINd field1 rel1 value1 [OR value1a OR value1b OR ...]

[{AND|,} field2 rel2 value2 {AND|,}]
```

where:

fieldn ...

Is a field in the segment.

reln ...

Is one of the following relations:

`EQ` or `=`	Equal to
`NE`	Not equal to
`GE`	Greater than or equal to
`GT`	Greater than
`LE`	Less than or equal to
`LT`	Less than
`CONTAINS` or `CO`	Contains the character string
`OMITS` or `OM`	Omits the character string

valuen ...

Is a value for which FSCAN can test. The first instance having the field value that passes the test becomes the current segment. If there are multiple tests, the first instance that passes all the tests becomes the current instance.

OR

Allows you to test a field for multiple values. If the field contains one of the values, it meets the test. You can use AND and OR in the same FIND command.

FIrst

Selects the first instance in a group of instances on display to be the current instance. In the root segment, the group of instances consists of all instances in the segment; in a descendant segment, a group consists of all instances that share a common parent instance.

FOrward

Scrolls the display one screen forward.

PF keys: PF8 or PF20.

Help

Invokes the FSCAN HELP facility.

PF keys: PF01 or PF11.

Input

Adds a new segment instance.

Prefix area command: `I`

Note: This command is valid only in the input area as a prefix command.

Jump

Moves to the child of the next parent instance.

PF keys: PF12 or PF24

LAst

Selects the last instance of a group of instances on display. In the root segment, the group of instances consists of all instances in the segment; in a descendant segment, a group consists of all instances that share a common parent instance.

LEft

Scrolls the display one panel to the left.

PF keys: PF10 or PF22.

LOcate

Searches for instances containing field values that fulfill certain conditions. LOCATE searches starting from the current instance. If it finds the instance, it makes that instance the current instance.

The syntax is (entered on one line)

```
LOcate field1 rel1 value1 [OR value1a OR value1b OR ...]

[{AND|,} field2 rel2 value2 {AND|,}]
```

where:

fieldn ...

Is a field to be tested.

reln ...

Is one of the following relations:

`EQ` or `=`	Equal to
`NE`	Not equal to
`GE`	Greater than or equal to
`GT`	Greater than
`LE`	Less than or equal to
`LT`	Less than
`CONTAINS` or `CO`	Contains the character string
`OMITS`	Omits the character string

or `OM`	

valuen ...

Is a value for which FSCAN can test. The first instance having the field value that passes the test becomes the current segment. If there are multiple tests, the first instance that passes all the tests becomes the current instance.

OR

Allows you to test a field for multiple values. If the field contains one of the values, it meets the test. You can use AND and OR in the same LOCATE command.

Key

Enables you to type over key field values in the current instance.

Prefix area command: `K`

where:

K/

Makes the instance the current instance after the key values are changed.

Multiple

Displays multiple instances, each on a single line. Entering this command after entering the SINGLE command returns the screen to the normal display (see Displaying a Single Instance on One Screen: The SINGLE and MULTIPLE Commands).

Next [n]

Scrolls the display *n* lines forward. *n* defaults to 1.

Parent

Displays the parent segment. The parent instance becomes the current instance. In SINGLE mode, PARENT displays the parent instance only.

QUit

Exits the FSCAN facility if you did not make any changes to the database.

PF keys: PF3 or PF15.

QQuit

Exits the FSCAN facility without saving any changes to the database.

REPlace

Replaces field values. The syntax is

```
REPlace field1 = value1[,field2 = value2 ...] [,$ {*|n}]
```

where:

fieldn...

Is a field in the instance whose value you want to change.

valuen...

Is the new value for the field.

,$ {*|n}

Enables you to change multiple instances counting from the current instance (the current instance included). *n* is the number of instances to be searched for the field values you want to change. If you want all instances in the group searched (starting from the current instance), use an asterisk (*).

You can also replace field values by typing over them.

REPlace KEY

Replaces key field values in the current instance. The syntax is

```
REPlace KEY key1 = value1[, key2 = value2, ...]
```

where:

keyn ...

Is a key field in the instance whose value you want to change.

valuen ...

Is the new value for the key field.

You can also replace key field values by typing over them.

RESet

Performs the following:

- Clears the input area.
- Recovers all field values on the screen that you typed over, both non-key fields and key fields. To recover non-key field values, you must enter the RESET command before you press the **Enter** key. Otherwise, you will not recover the typed-over values.

PF keys: PF2 or PF14.

Prefix area command: `R`

Note: The R prefix-area command recovers only field values on the line that it is typed. If you typed changes on a line not specifying the R prefix, FSCAN enters the changes.

RIght

Scrolls the display one panel to the right.

PF keys: PF11 or PF23.

SAve

Saves all changes made to the database without exiting FSCAN.

SIngle

Displays the current instance alone with all field values on one screen. To return to the normal display, enter the MULTIPLE command.

Top

Displays the root segment and makes the first instance in the root segment the current instance, scrolled to the leftmost panel.

?

Displays the previous command in stack.

PF keys: PF6 or PF18.

=

Executes the previous command entered.

PF key: PF9 or PF21.

Summary of PF Keys

The following table is a list of FSCAN PF keys and their corresponding functions.

FSCAN Keys	Functions
PF1, PF13	HELP
PF2, PF14	RESET
PF3, PF15	QUIT
PF4, PF16	PARENT
PF5, PF17	CHILD
PF6, PF18	?
PF7, PF19	BACKWARD
PF8, PF20	FORWARD
PF9, PF21	=
PF10, PF22	LEFT
PF11, PF23	RIGHT
PF12, PF24	JUMP

Summary of Prefix Area Commands

The following is a summary of prefix area commands. You type these commands in the prefix area that corresponds to the instance you wish to address.

/	Makes the instance the current instance. May be typed after the prefix area commands K, I, and R.
C	Displays child instances (see Displaying Descendant Segments: The CHILD, PARENT, and JUMP Commands).
D	Deletes the instance and all its children.
I	Inputs a new instance (valid only in the input area).
I/	Inputs a new instance and makes the instance the current instance (valid only in the input area).
K	Enables you to type over key field values in the instance.
K/	Enables you to type over key field values in the instance, then makes the instance the current instance.
R	Performs the following: • Clears the input area. • Recovers all field values on the screen that you typed over, both non-key fields and key fields. To recover non-key field values, you must enter the RESET command before you press the Enter key. Otherwise, you will not recover the typed-over values. Note that the R prefix area command recovers only field values on the line on which it is typed. If you typed changes on a line not specifying the R prefix, FSCAN enters the changes.

ibi Documentation and Support Services

For information about this product, you can read the documentation, contact Support, and join Community.

How to Access ibi Documentation

Documentation for ibi products is available on the Product Documentation website, mainly in HTML and PDF formats.

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the ibi™ FOCUS® Documentation page.

How to Contact Support for ibi Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, you can request one by clicking **Register** on the website.

How to Join ibi Community

ibi Community is the official channel for ibi customers, partners, and employee subject matter experts to share and access their collective experience. ibi Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from ibi products. For a free registration, go to ibi Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ibi, the ibi logo, FOCUS, iWay, WebFOCUS, RStat, Information Builders, Studio, and TIBCO are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.