UNISYS

ALLY® Software Development Environment Utilities User's Guide

June 1988

Printed in U S America

Priced Item

UP–12508 Rev. 1

Preface

This manual describes ALLY release 2.0.

This manual contains information for developers on how to use ALLY utilities. ALLY utilities allow you to manage and maintain your application's AFILEs, data, and macro command files.

This manual contains eight chapters and three appendixes.

Chapter 1 gives an overview of the ALLY utilities, introduces invocation methods, discusses error messages, and describes the utilities' relationship to the Format File and the symbol table.

Chapter 2 describes the AFILE Compactor, which optimizes and removes unused space from AFILEs.

Chapter 3 discusses the AFILE Merger, which merges information from two AFILEs.

Chapter 4 discusses the AFILE Message Builder, which creates library help and error message AFILEs, merges message files into your application AFILE, and unloads a message AFILE or the messages from the application AFILE into a portable file.

Chapter 5 discusses the AFILE Script Writer, which produces a text description of application AFILEs.

Chapter 6 describes the AFILE Migrator, which allows you to transport your applications to other operating systems.

Chapter 7 describes the Data Migrator, which allows you to transport your application's data to other operating systems and access methods.

Chapter 8 describes the Macro Utility, which allows you to transport ALLY command files to other operating systems.

Appendix A contains a summary of the command lines used to invoke ALLY's utilities.

Appendix B lists the mnemonics for all ALLY commands. These mnemonics are used in Macro Utility text files.

Appendix C contains a table of the ASCII character set with decimal, hexidecimal, and octal codes.

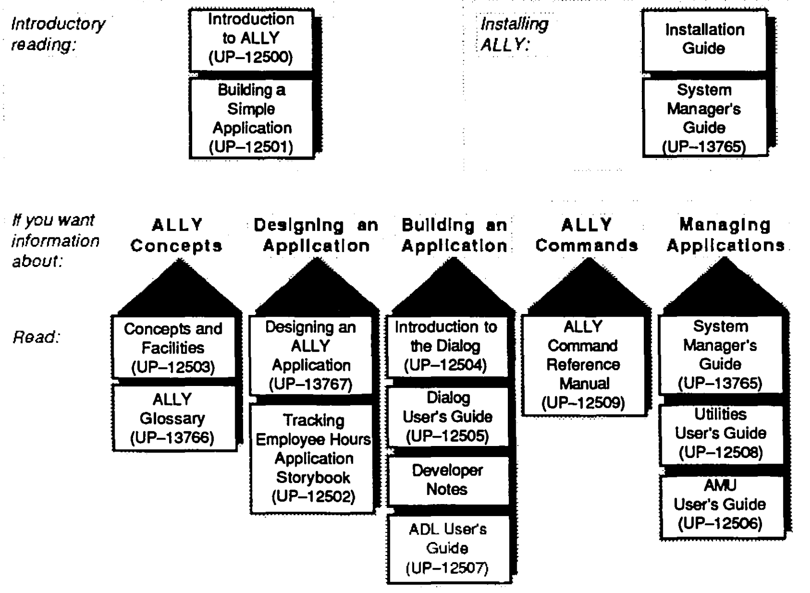
The ALLY Documentation—What to Read

The following illustration shows you how the ALLY documentation is organized.

If you want to develop ALLY applications, we suggest that you start by reading the ALLY system's introductory brochure, *Introduction to ALLY*. Then, you can build the application in *Building a Simple Application*.

If you want to install ALLY, you should read the installation guide for your system.

Note that the documentation for the runtime version of ALLY includes only the installation guide for your system, *ALLY Command Reference Manual*, and *AMU User's Guide*.



Introductory reading:
Introduction to ALLY (UP-12500)
Building a Simple Application (UP-12501)
Installing ALLY:
Installation Guide
System Manager's Guide (UP-13765)
If you want information about:
Read:
ALLY Concepts
Concepts and Facilities (UP-12503)
ALLY Glossary (UP-13766)
Designing an Application
Designing an ALLY Application (UP-13767)
Tracking Employee Hours Application Storybook (UP-12502)
Building an Application
Introduction to the Dialog (UP-12504)
Dialog User's Guide (UP-12505)
Developer Notes
ADL User's Guide (UP-12507)
ALLY Commands
ALLY Command Reference Manual (UP-12509)
Managing Applications
System Manager's Guide (UP-13765)
Utilities User's Guide (UP-12508)
AMU User's Guide (UP-12506)

F002-0814-00

Conventions

You should read carefully the description of documentation conventions before reading this manual.

We use the following conventions in this manual:

Single quotes (‘ ’)	Identify command names.
Boldface type (**bold**)	Highlights text you are to enter. Boldface is also used within command syntax statements.
Double quotes (“ ”)	Identify text strings within text sections. These strings are typically located in examples or as part of the prompts that ALLY sends to your display. Sometimes the exact content of a text string is affected by the traditional rules of punctuation. In these cases, we place the closing quotation mark at the end of the text string. For example, instead of: You see the prompt “Macro number:.” We say: You see the prompt “Macro number:”.
Brackets (<x>)	Enclose a specific key (x = key) to be typed. Angle brackets are used in command syntax and key assignment lines. For example: <,> the “comma” key on the numeric keypad <Return> the “Return” key <Do> the “Do” key <F18> the “function” key 18 <2> the “2” on the numeric keypad
Square brackets ([])	Enclose an argument for the command-line invocation of a utility. For example, **newprint** *[printer name] [output file]*.

End of Preface

Contents

Chapter 5. The AFILE Script Writer

Chapter 6. The AFILE Migrator

Chapter 7. The Data Migrator

Chapter 8. The Macro Utility

Figures

Tables

Chapter 1
Introduction

The ALLY Utilities

ALLY provides seven utilities for managing, maintaining, and transporting application AFILEs, data, and macro command files. These utilities are the:

- AFILE Compactor
- AFILE Merger
- AFILE Message Builder
- AFILE Script Writer
- AFILE Migrator
- Data Migrator
- Macro Utility

This manual describes in detail each of these utilities. ALLY also provides two additional utilities, the Printer Definer and the Terminal Definer, that build printer and terminal description files. See the *System Manager's Guide* (UP-13765) for information about the Printer Definer and Terminal Definer utilities.

The following sections discuss characteristics that are common to all ALLY utilities: invocation, environment variables, the Format File, error messages, and symbol tables.

Invoking the Utilities

You can invoke ALLY utilities from the Application Developer's Dialog, commonly called the Dialog, or from your operating system's command line. The Dialog allows you to invoke the utilities from its application maintenance branch. Utilities invoked from your operating system's command line accept one or more arguments. Appendix A lists these operating system commands and their arguments.

The way you specify arguments can be operating-system dependent. "Help" is a reserved word used to get the syntax for invoking the utilities from the operating system's command line. Enter the command followed by "help" to display syntax information for your operating system. For example,

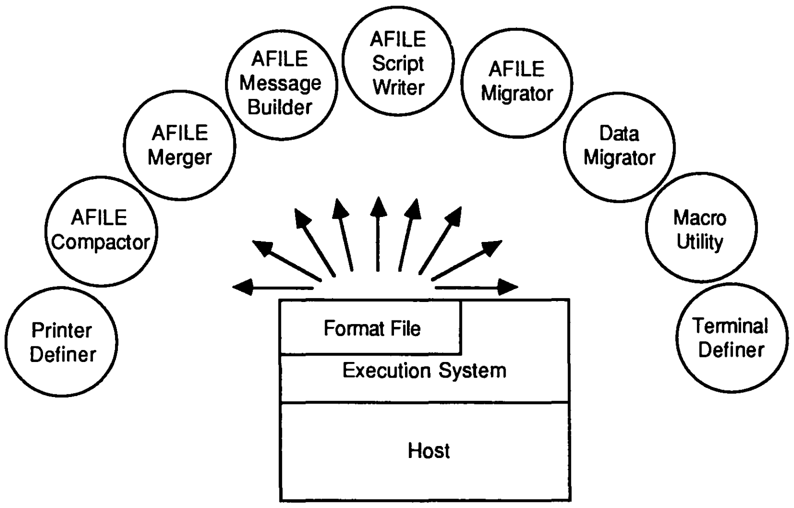
```
acompact help<Return>
```

displays the syntax for the script that invokes the AFILE Compactor utility.

You can also invoke a subset of the utilities from the Application Maintenance Utilities (AMU). The AMU is designed to be used with the runtime-only version of ALLY. For more information about the AMU, see the *Application Maintenance Utilities (AMU) User's Guide* (UP-12506).

The Format File

All ALLY utilities use information from the Format File, which is part of ALLY, to display text. The Format File also contains a pointer to ALLY's error AFILE and the default help number for error messages built with the AFILE Message Builder. Each utility accesses the Format File section it needs to execute and to produce messages.

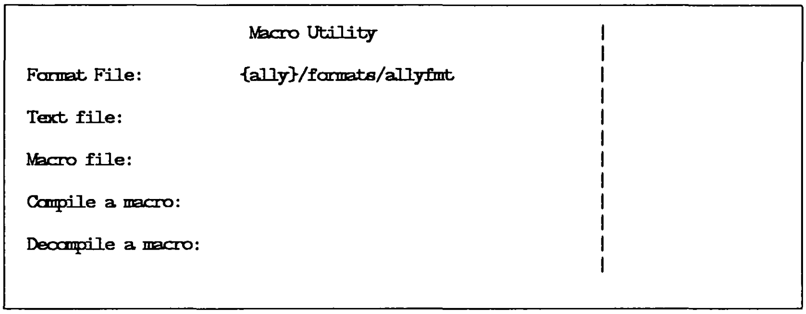


F002-0579-01

Figure 1-1. The Format File and the Utilities

The first entry in the Format File must point to an error AFILE. The error AFILE contains the text strings the utilities need to produce warning and error messages. The Format File is essential for the utilities to execute. If a utility cannot locate a valid Format File, it displays an error message and stops processing.

If your system manager has correctly set the ALLY environment variable, the name of the Format File currently being used is already filled in for you on each utility's initial Dialog form, as shown in Figure 1-2. Type <Return> to use the Format File that is displayed. To use a different Format File, either edit or delete the name that is displayed and type the name of the Format File that you want to use. If you enter a name that is not a valid Format File, you receive an error message that says the Format File cannot be opened.



```
                         Macro Utility
Format File:             {ally}/formats/allyfmt
Text file:
Macro file:
Compile a macro:
Decompile a macro:
```

Figure 1-2. Format File Name in Dialog Menus

Another way to use a different Format File is to copy the current Format File to another file and then rename the Format File. (Copying the Format File preserves the original Format File.) The *System Manager's Guide* (UP-13765) contains a detailed discussion of the Format File supplied with ALLY. It also tells how to use a different Format File when you invoke the utilities from your operating system's command line.

Error Messages from the Format File

If the Format File does not contain a pointer to a valid error file, the utilities can still execute and produce some error and warning messages. If an error is not among those in the following list, the utilities display only the number of the error.

The utilities return an error when:

- a path name to the error file is incomplete
- a Format File cannot be opened
- a Format File name is not valid
- an option is invalid
- a file cannot be opened
- a file cannot be located

The utilities produce a warning message when an error condition is not severe enough to prevent processing.

The Symbol Table

When an ALLY application is defined, a symbol table is created as an internal part of the application's AFILE. The symbol table maintains these parts of an application:

- AFILE item names
- AFILE item types
- AFILE item locations
- the relationships between AFILE items

The AFILE Compactor allows you to move the symbol table from your application AFILE to an external file. Although moving the symbol table to an external file decreases the size of your application AFILE, it does not affect the use of memory during execution.

Removing the symbol table from your application AFILE also provides a mechanism for security. When the symbol table file is renamed or moved from the AFILE's directory, the AFILE cannot be modified with the Dialog, AMU, or utilities.

You can also use the AFILE Compactor to move an external symbol table file back into the AFILE. See Chapter 2 for details on moving an external file back into an AFILE.

The AFILE Compactor, AFILE Message Builder, AFILE Merger, and AFILE Migrator utilities allow you to create an external symbol table file for the new AFILE that the utility produces. The utilities can access an external symbol table file as long as the file has not been moved from the AFILE's directory, deleted, or renamed.

Because the default symbol table file is "none," which means that the AFILE has an internal symbol table, you cannot use "none" or "NONE" as the name of an external symbol table file.

Environment Variables

You can assign parts of your ALLY application to a name that is an environment variable in your operating system. ALLY looks for the name's value when the application executes. Using an environment variable allows you to specify a short variable name instead of a long path name to locate a file, like the Format File. Using environment variables can also make your applications more portable across operating systems by minimizing changes due to syntax differences.

You can specify environment variables for the following parts of your application:

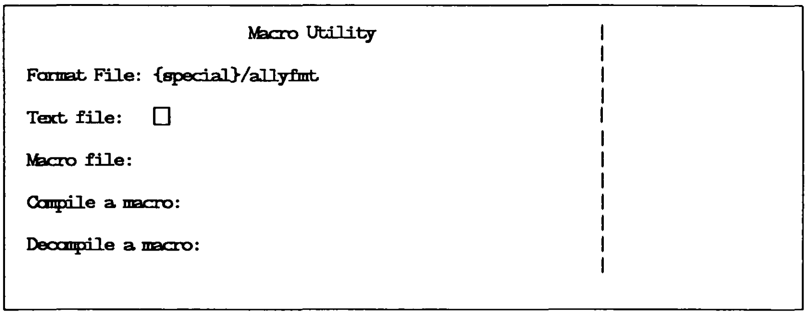
- utility files
- help and error library AFILEs
- some data source files (see the developer notes for your access methods)

Two operating system environment variables are supplied with ALLY. One is "ally." Its value points to the top-level ALLY directory. The default Format File and help and error library AFILEs use the "ally" environment variable. The other environment variable, "allyprinter," can override the default printer spooling device or queue name. The environment variable names "ally" and "allyprinter" are reserved, so you cannot change their names.

You can also define your own operating system environment variables for use in your application. You define environment variables with an operating system command before your ALLY application executes.

ALLY recognizes a name enclosed in braces as an operating system environment variable, and substitutes its value from the environment variable list. Environment variables used as defaults appear in braces ({*name*}) in the Dialog's forms. To specify an environment variable that you have defined, enclose the name in braces in the field of the appropriate form.

Figure 1-3 shows the environment variable, "special," specified for the Format File. Because "special" is enclosed in braces, ALLY knows that it is an operating system environment variable, in this case, defined as the path to the Format File. In this example, ALLY uses the "special" environment variable to access a Format File stored in a directory of files translated to a local language. At another time, "special" might be redefined to point to the directory of files translated to a different language.



```
                    Macro Utility

Format File: {special}/allyfmt

Text file:  □

Macro file:

Compile a macro:

Decompile a macro:
```

Figure 1-3. Using an Environment Variable

Displaying Environment Variables

Table 1-1 lists the commands used by some operating systems to display environment variables. If your system is not listed here, see the installation guide for your system.

Table 1-1. Commands to Display Environment Variables

Operating System	Command
MS-DOS	set
UNIX Berkeley 4.x	printenv
UNIX System 5	env

Defining Environment Variables

Table 1-2 lists the commands for some operating systems to define an environment variable. If your system is not listed here, see the installation guide for your system.

Table 1-2. Commands to Define Environment Variables

Operating System	Command
MS-DOS	set *variable=value*
UNIX C-shell	setenv *variable value*
UNIX Bourne shell	*variable=value* export *variable*

The MS-DOS "set" command defines a variable for the duration of a PC session. If you include the "set *variable=value*" command in your "autoexec.bat" file, the variable is defined for every session.

UNIX users can place the appropriate command in the ".cshrc" (C-shell) file or ".login" (Bourne shell) file in their home directory.

Summary

The ALLY utilities allow you to manage AFILEs and transport applications. You can run these utilities from the Dialog or from your operating system's command line. All utilities require a valid Format File to operate. The Dialog menus have the current Format File name filled in for you. When you execute the utilities from your operating system's command line, the command file automatically uses the current Format File.

End of Chapter 1

Chapter 2 The AFILE Compactor

Introduction

The AFILE Compactor utility (Figure 2-1) compacts an AFILE and produces a new, condensed AFILE. The compacted AFILE is upgraded to the current version number if the input AFILE is a lower version.

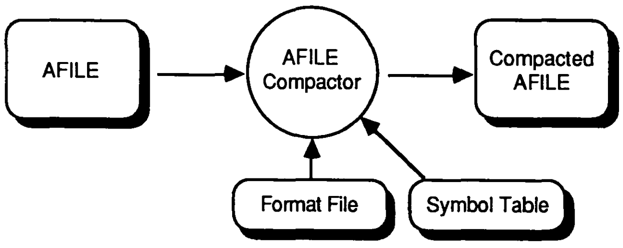


Figure 2-1. The AFILE Compactor

Changing an AFILE often increases its size by leaving unused items in the AFILE. The AFILE Compactor (the Compactor) removes unused items and optimizes the arrangement of the AFILE entries in the compacted AFILE by putting related parts of an application (e.g., the description of a menu, its text, and its choices) on the same or nearby disk blocks. The Compactor minimizes the number of AFILE disk blocks that ALLY must read as the application executes.

You can decrease the size of your application's AFILE by using the Compactor to move the symbol table to an external file. (If you run your application on a PC, the symbol table should be in an external file to save space.) You can also move an external symbol table file back into the AFILE with the Compactor utility.

Be careful with external symbol table files. If you lose or damage an AFILE's external symbol table file, you cannot make modifications to the AFILE. You can control modifications to your application by renaming or moving an external symbol table file out of the AFILE's directory, because the AFILE cannot be modified without the symbol table file. If you rename the symbol table file, you must restore its original name before you can run the AFILE. To run an AFILE with an external symbol table in a different directory, you must use an operating system command to establish a link between the two directories. On UNIX systems you use the "link" command and on the PC you use the MS-DOS "set path" command.

You can run the AFILE Compactor at the end of every Dialog session, or periodically as desired. However, you do not need to compact an AFILE before you execute it.

Options for the AFILE Compactor

The AFILE Compactor has three options that are summarized in Table 2-1.

Table 2-1. AFILE Compactor Options

Option	Action
Delete disjoint items	Deletes from the compacted AFILE all items that are not referenced by another item in the AFILE.
Suppress statistics	Does not produce a list of statistics about the compacted AFILE.
Use virtual maps	Uses virtual maps when a system has limited main memory.

Delete Disjoint Items

A disjoint item is an AFILE item that is not referenced by another item in the AFILE. For example, a menu that has been defined but never pointed to by an AFILE item is a disjoint item. By default, every item in the named AFILE (including disjoint items) is put into the new, compacted AFILE.

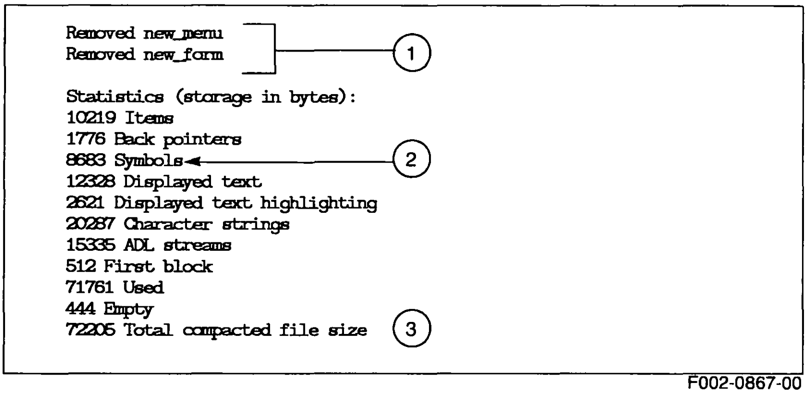
Often, disjoint items in your AFILE are unused items left from previous development sessions. The Compactor does not copy disjoint items to the new AFILE it creates if you specify the option to delete disjoint items.

If you use this option and inadvertently delete an item that you later need from the AFILE, you can use the Dialog to copy that item from the old AFILE to the compacted one. To avoid inadvertent deletions, delete disjoint items only after an AFILE has been tested and is ready to go into production.

Suppress Statistics

The Compactor prints, by default, a list of the space occupied by elements in the AFILE.

Figure 2-2 shows an example of the statistics that are displayed on your terminal if you do not specify the option to suppress statistics.



```
Removed new_menu
Removed new_form

Statistics (storage in bytes):
10219 Items
1776 Back pointers
8683 Symbols
12328 Displayed text
2621 Displayed text highlighting
20287 Character strings
15335 ADL streams
512 First block
71761 Used
444 Empty
72205 Total compacted file size
```

F002-0867-00

Figure 2-2. Compactor Statistics

The statistics reported are mainly of use to your ALLY distributor in diagnosing AFILE problems. Some of these statistics are:

1) A list of disjoint items that have been deleted.
2) The number of bytes consumed by symbol table information (symbols).
3) The number of bytes occupied by the entire AFILE (total compacted file size). If the symbols occupy a large percentage of the total AFILE size, you can use the Compactor to decrease the AFILE's size by moving the symbols to an external symbol table file.

Use Virtual Maps

Use virtual maps when your system has limited main memory. The Compactor runs more slowly when using virtual maps because memory space is preserved at the expense of execution time.

Invoking the AFILE Compactor from the Dialog

Figure 2-3 shows the location of the Dialog forms and subforms from which you can invoke the AFILE Compactor utility.

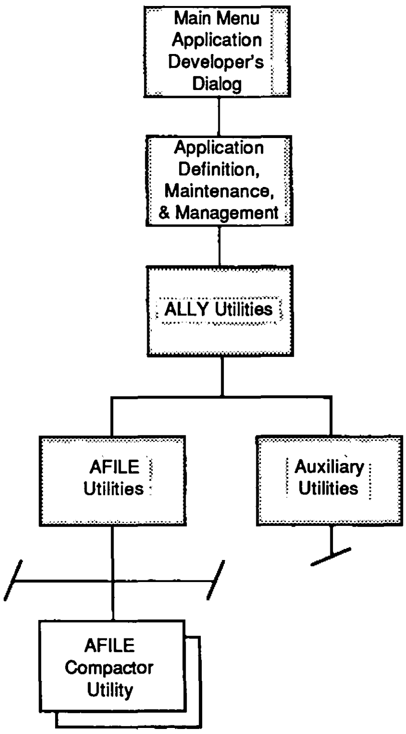
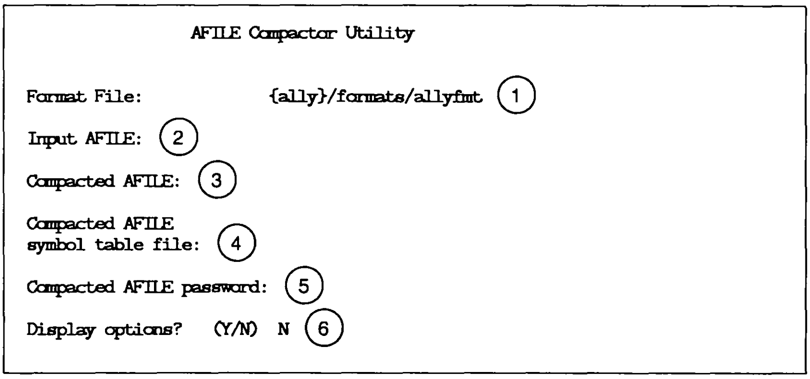

Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
AFILE Compactor Utility
F002-0811-00

Figure 2-3. Dialog Path to the AFILE Compactor

Figure 2-4 shows the Dialog form that you use to invoke the Compactor.



```
                    AFILE Compactor Utility

Format File:               {ally}/formats/allyfmt  (1)

Input AFILE:  (2)

Compacted AFILE:  (3)

Compacted AFILE
symbol table file:  (4)

Compacted AFILE password:  (5)

Display options?  (Y/N)  N  (6)
```

Figure 2-4. Invoking the AFILE Compactor

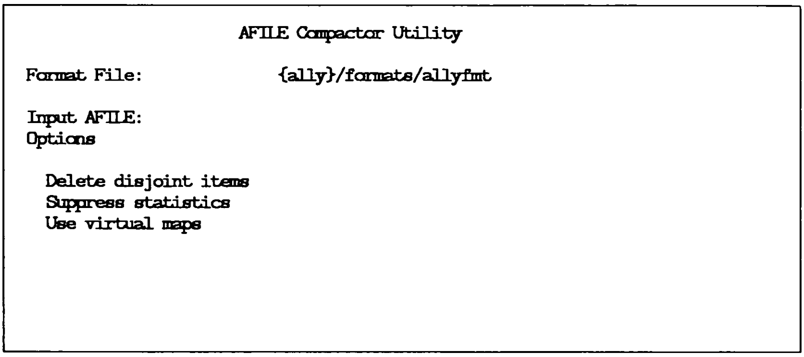
1) The name of the current Format File is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

Enter the following information in the remaining fields:

2) The name of the AFILE that you want to compact.
3) The name of the compacted AFILE produced. This name cannot be the same as the AFILE being compacted unless your operating system allows multiple versions of the same file.
4) The name of the external symbol table file, if you want to create one for the compacted AFILE. Type "none" if you do not want an external symbol table file or if you want to move an external symbol table file back into the input AFILE.
5) The password assigned to the AFILE you are compacting, or <Return> if the AFILE is not password-protected. Characters that you type in this field are not displayed. By default, an AFILE is not password-protected.

6) Type <Return> if you do not want to select a Compactor option. The cursor moves to the confirmation field.

   Type "Y<Return>" or "X<Return>" to select an option. The cursor moves to a subform (Figure 2-5) that lists the options.



```
                    AFILE Compactor Utility

Format File:                {ally}/formats/allyfmt

Input AFILE:
Options

  Delete disjoint items
  Suppress statistics
  Use virtual maps
```

Figure 2-5. AFILE Compactor Options

Type "Y<Return>" or "X<Return>" in the field for any option you want to select. After you enter the information required and use the 'exit action' command, the Compactor executes. While the Compactor runs, the cursor moves to the lower-left corner of the form. After processing is completed, the statistics report is displayed. When you issue the 'abort action' command, the cursor moves to the menu titled *AFILE Utilities*.

Invoking the AFILE Compactor from the Command Line

The command line to invoke the AFILE Compactor is:

acompact *[input AFILE] [output AFILE] [symbol table file] [password] [options]*

Only the first two arguments are required. The rest of the arguments default to:

- symbol table file = none (an internal symbol table)
- password = none (not password-protected)

Here is a description of the arguments:

input AFILE	The name of the AFILE to be compacted.
output AFILE	The name of the compacted AFILE produced. This name cannot be the same as the AFILE to be compacted unless your operating system allows multiple versions of the same file.
symbol table file	The external symbol table file for the new AFILE. Specify "none" if you do not want an external symbol table file or if you want to move an external symbol table file back into the input AFILE.
password	The password of the AFILE to be compacted, or "none" if the AFILE is not password-protected. By default, an AFILE is not password-protected.
options	If desired, one or more of the options listed below. See the "Options for the AFILE Compactor" section for details.

d	delete disjoint items
s	suppress statistics
v	use virtual maps.

Invocation Example

```
acompact oldsales.a newsales.a none none dv
```

This example compacts an AFILE named "oldsales.a", which does not have an external symbol table and is not password-protected. The compacted output file is named "newsales.a". The "d" and "v" options delete disjoint items and use virtual maps.

AFILE Compactor Error Messages

The AFILE Compactor produces an error message if it finds:

- an incorrect number of command line arguments
- an invalid password
- file name conflicts

End of Chapter 2

Chapter 3
The AFILE Merger

Introduction

The AFILE Merger allows you to merge information from a master AFILE and a second AFILE (Figure 3-1) The information is either merged into the existing master AFILE or written to a new AFILE. You can use the AFILE Merger to add new information to an application or to join two separate applications.

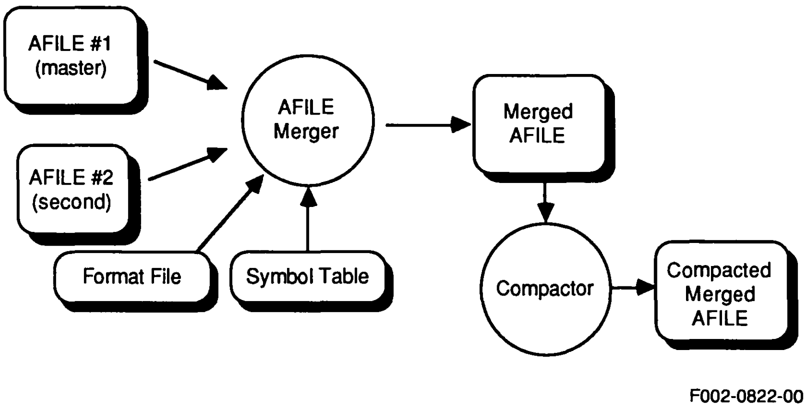


Figure 3-1. The AFILE Merger

The Merger allows you to designate which of two AFILEs is to be merged into the other. You will often use the Merger to add to a master AFILE from a second AFILE that is undergoing simultaneous development. By default, an item from the second AFILE always takes precedence when identically named items exist in both AFILEs. You can change this by selecting an option that gives the master AFILE precedence in any item-name conflicts.

By default, the Compactor automatically compacts AFILEs that the Merger creates. You can select an option to suppress automatic compaction. The Merger calls the Compactor before merging if one of the AFILEs has an internal symbol table and the other has an external symbol table file. You cannot suppress this special Compactor invocation.

How the AFILE Merger Works

The Merger utility has five steps:

1) The Merger copies the master AFILE into the new, merged AFILE. This step is not done when you specify "none" as the name of the merged AFILE because you are merging the second AFILE into the master AFILE.
2) The Merger compares the global information items (not item names) in each AFILE. The Merger displays an error message if there are differences between global information items. Each global information item that does not exist in the new AFILE is copied from the second AFILE. This part of the merge operation continues until all global information items in each AFILE have been compared.
3) The Merger copies the AFILE items from the second AFILE to the master. During this phase, the Merger merges into the new AFILE each item from the second AFILE.
4) The Integrity Reporter verifies all of the actions in the new AFILE. A diagnostic message is displayed if an error is found.
5) In this optional step, the Compactor utility compacts the new AFILE. You can choose to have the Compactor:
    - give you statistics for the new AFILE
    - remove disjoint items from the new AFILE

Options for the AFILE Merger

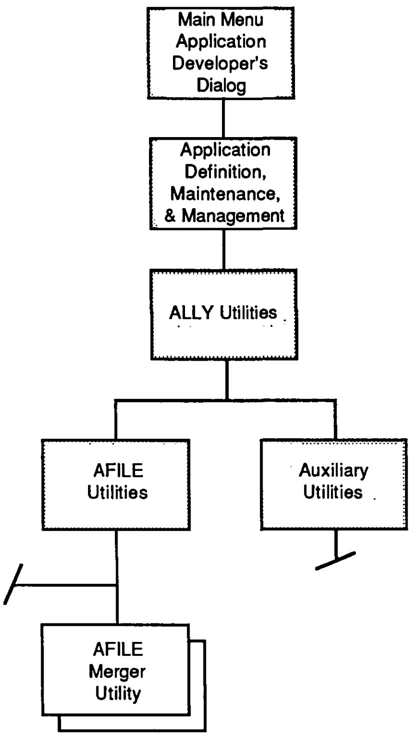
Table 3-1 lists the seven options that are available for the AFILE Merger.

Table 3-1. AFILE Merger Options

Option	Action
Delete disjoint items with the AFILE Compactor	Deletes AFILE items not invoked by an item in the new AFILE.
Suppress statistics from the AFILE Compactor	Does not produce Compactor statistics.
Suppress compaction of merged AFILE	Does not compact the new AFILE.
Suppress entry point merge	Does not copy entry points from the second AFILE to the new AFILE.
Suppress global variable and list merge	Does not copy global variables from the second AFILE to the new AFILE.
Resolve conflicts with item from the master AFILE	Copies an item from the master AFILE when there is a name conflict.
Use virtual maps	Uses virtual maps when your system has limited main memory.

Invoking the AFILE Merger from the Dialog

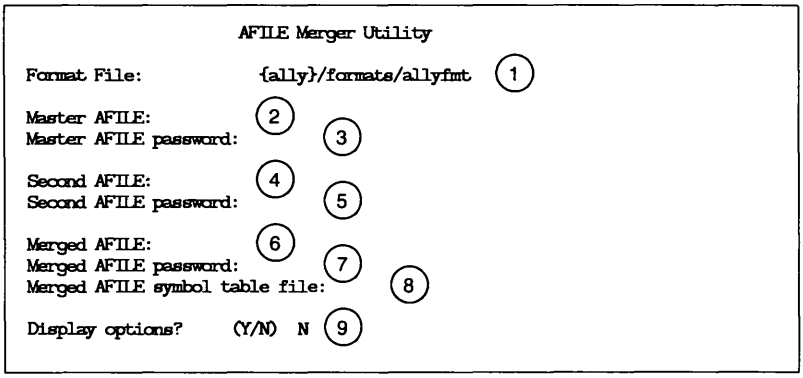
Figure 3-2 shows the location of the Dialog forms and subforms from which you can invoke the AFILE Merger utility.


Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
AFILE Merger Utility

F002-0813-00

Figure 3-2. Dialog Path to the AFILE Merger

Figure 3-3 shows the Dialog form that you use to invoke the AFILE Merger.



```
                        AFILE Merger Utility

Format File:            {ally}/formats/allyfmt  (1)

Master AFILE:           (2)
Master AFILE password:      (3)

Second AFILE:           (4)
Second AFILE password:      (5)

Merged AFILE:           (6)
Merged AFILE password:      (7)
Merged AFILE symbol table file:  (8)

Display options?     (Y/N)  N  (9)
```

Figure 3-3. Invoking the AFILE Merger

1) The current Format File name is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

Enter the following information in the remaining fields:

2) The name of the master AFILE for the merge operation.

3) The password for the master AFILE or <Return> if the AFILE is not password-protected. Characters that you type in this field are not displayed. By default, an AFILE is not password-protected.

4) The name of the second AFILE for the merge operation.

5) The password for the second AFILE or <Return> if the AFILE is not password-protected. Characters that you type in this field are not displayed. By default, an AFILE is not password-protected.

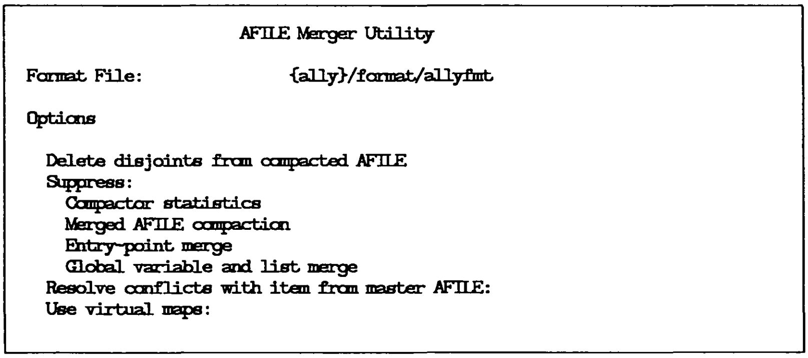
6) The name of the AFILE to be created by the merge operation. Type <Return> to merge the second AFILE into the master AFILE instead of creating an output file. This name cannot be the same as the second AFILE or the same as the master AFILE unless your operating system allows multiple versions of the same file.

7) The password for the new, composite AFILE or <Return> if you do not want to assign a password to the new AFILE. Characters that you type in this field are not displayed. By default, an AFILE is not password-protected.

8) The name of the external symbol table file if you want to create one for the composite AFILE. If you do not want an external symbol table file, type <Return>.

9) Type <Return> if you do not want to select an option. The cursor moves to the confirmation field.

   Type "Y<Return>" or "X<Return>" if you want to select an option. The cursor moves to a subform Figure 3-4) that lists the options.



```
                        AFILE Merger Utility

Format File:                    {ally}/format/allyfmt

Options

  Delete disjoints from compacted AFILE
  Suppress:
    Compactor statistics
    Merged AFILE compaction
    Entry-point merge
    Global variable and list merge
  Resolve conflicts with item from master AFILE:
  Use virtual maps:
```

Figure 3-4. AFILE Merger Options

Type "Y<Return>" or "X<Return>" next to any option you want to select. After you enter the information and use the 'exit action' command, the Merger executes. The cursor moves to the lower-left corner of the form while the Merger runs. After processing is completed, the statistics report is displayed. When you issue the 'abort action' command, the cursor moves to the menu titled *AFILE Utilities*.

Invoking the AFILE Merger from the Command Line

The command line for invoking the AFILE Merger is:

```
amerge [master AFILE] [master AFILE password] [second
AFILE] [second AFILE password] [output AFILE] [output
AFILE password] [symbol table file] [options]
```

The operating-system command file for this utility automatically uses the current Format File.

On the command line, you must specify for the arguments:

master AFILE	The name of the master AFILE (to merge into).
master AFILE password	The password for the master AFILE. The default password is "none," which means that the AFILE is not password-protected.
second AFILE	The name of the second AFILE from which to take items.
second AFILE password	The password for the second AFILE, or "none" if the AFILE is not password-protected.
output AFILE	The name of the new AFILE to be created from the master and second AFILEs. Specify "none" to merge the second AFILE into the master AFILE. This name cannot be the same as the second AFILE or the same as the master AFILE unless your operating system allows multiple versions of the same file.
symbol table file	The name of the external symbol table file for the composite AFILE. Specify "none" if you do not want an external symbol table file.

output AFILE password	The password for the AFILE to be created, or "none" if the AFILE is not password-protected.
options	If desired, one or more of the following:

c	suppress compaction of the new AFILE
d	delete disjoint items
e	suppress entry point merge
g	suppress global variable and list merge
m	resolve conflicts with item from master AFILE
s	suppress statistics from the AFILE Compactor
v	use virtual maps.

Invocation Example

```
amerge big.a look small.a none both.a locked none dvm
```

This example merges a master AFILE named "big.a" with another AFILE, named "small.a". The composite output file is an AFILE named "both.a". The master AFILE (big.a) is protected with the password "look". The output AFILE (both.a) does not have an external symbol table file but is to be protected with the password "locked". The "d," "v," and "m" options are specified to delete disjoint items, use virtual maps, and resolve conflicts with master AFILE items.

AFILE Merger Error Messages

The AFILE Merger produces an error message if you:

- specify an invalid password for either input AFILE.
- specify the name of the new external symbol table file that is the same as the external symbol table file of the master or second AFILE.
- try to merge an AFILE into a master AFILE with a lower version number. (The AFILE Merger does not produce an error if you merge into a new output AFILE.)

The Merger can report that there are integrity errors. You can read these integrity error messages by invoking 'start task 2' and selecting choice 2, "AFILE item information." Then select the item for producing integrity reports on the entire AFILE.

End of Chapter 3

Chapter 4
The AFILE Message Builder

Introduction

The AFILE Message Builder allows you to:

- create library AFILEs for help and error messages
- merge a message file into your application AFILE or an existing library AFILE
- unload a message AFILE, or the messages from an application AFILE, into an ASCII text file

If your application is very large, you may want to store the text of help and error messages in library AFILEs to keep the application AFILE as small as possible.

Figure 4-1 shows the basic operation of the Message Builder.

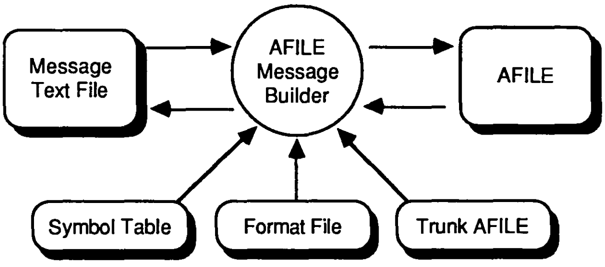


Figure 4-1. The AFILE Message Builder

The Message Builder uses an ASCII text file to build a message AFILE. This text file can contain help, legend, or error messages that can be edited with the ALLY Text Editor (or any standard editor), or formatted with a text formatter. This text file also contains special strings, called directives, that control highlighting and signal to the Message Builder the beginning and end of each message. Message Builder directives are defined in the Format File.

Building a Message Text File

Text files for the Message Builder contain messages, each of which must be surrounded with directives that specify the message type and start and end of the message. These message type directives begin and end with a dollar sign ($) and are listed in Table 4-1. Each directive (except the continuation symbol and the highlighting directives) must start in column one of a new line. The first characters after the "start" directives ("e," "h," and "l") must be a number assigned as the error, help, or legend number for this part of the application when it was built with the Dialog.

When you build an error or help AFILE for your application, you can include two special directives that signify the path and file name to additional error and help AFILEs. If you do not include these directives, your message AFILE will get default paths to ALLY's general help and error AFILEs. Refer to the *Dialog User's Guide* (UP-12505) for information on how to establish pointers in your application AFILE to application help and error AFILEs.

The Message Builder requires the message text file to be in a special format. Figure 4-2 shows part of a sample help message file. After the example, the labeled parts that illustrate the rules for Message Builder text files are explained.

```
①
   $efn$ {ally}/afiles/errors/errors.e
②  $hfn$ {ally}/afiles/common/common.h

③  $h$ 120 130 140 150 160 170 \ ④
   180 190 200
⑤  $help$ 900
            Help for Main Menu
    The choices on this menu allow you to define
    this application's menus, forms/reports, tasks,
    and procedural languages.  You can specify the
    data- and application-related information.  You    ⑥
    can also access the form that allows you to
    execute this application, and the help message
    about the Dialog and its help facility.
⑦  $*$
```

Figure 4-2. Sample Message Text File

In the sample message text file shown above:

1) The directive (efn) gives the path and file name of an error AFILE. When ALLY is shipped, the path and file name for the error AFILE is as shown for UNIX systems.

2) The directive (hfn) gives the path and file name of a help AFILE. When ALLY is shipped, the path and file name for the help AFILE is as shown.

3) The directive (h) signals the beginning of the message and must be followed by one or more numbers. The number immediately following the "h" is the help number assigned to this part of the application when it was built. You can assign any number between 1 and 4999 to help messages. (You can assign any number between 1 and 8191 to error messages.) The help number can be assigned to a call packet, a form/report field, a menu, or menu choice.

The optional numbers that follow (followed by any separator, including <Space>) are the numbers of any other items that also use this help message. For example, you can use the same help message for several fields in a form. A message attached to many different fields must have all its help numbers specified on the first line, unless a continuation character (\) is present at the end of the first line. Do not specify duplicate message numbers.

4) The optional continuation character (\) continues the message numbers on the next line. If you do not use a continuation character in the first line, numbers on any line after the first are treated as part of the message text. Continued lines can have a maximum of 512 characters.

5) The optional directive ($help$) signals that there is a second-level help message (here numbered 900) associated with this help message. When a user types 'help' from the first-level help message, the text of the second-level message appears. If you do not have a second-level help message, you do not need this directive line in your help message text list.

6) The message text for the specified message number can be any length and is displayed in the coordinates you designate for help and error messages. The text is displayed exactly as it is entered in the input text file, except that the directives and control characters (other than "tab" and "newline") are removed.

7) The directive ($*$) signals the end of a message.

It is a good idea to limit the length of all lines to eighty characters to eliminate difficulty when:

- working with text editors that have an eighty-column length limitation
- transporting message files to different operating systems

If your lines exceed eighty columns, you can use the option to wrap output text when unloading the message AFILE if you need to transport it to a different computer or use a text editor with an eighty-column length limitation.

Table 4-1. Message Type Directives

Directive	Purpose
h	Start of help message
l	Start of legend message
e	Start of error message
s	Start of embedded string definition
\	Line continuation character
$help$	Optional second level help message
$*$	End of each message

Highlight and Line-Draw Directives

There are Message Builder directives that turn highlighting on or off. You can choose to highlight any part of a message with:

- underlining
- reverse video
- blinking
- altered intensity
- any combination of the above

The Message Builder translates the highlighting so that it is not carried into margins and indentations when the highlighted string spans lines.

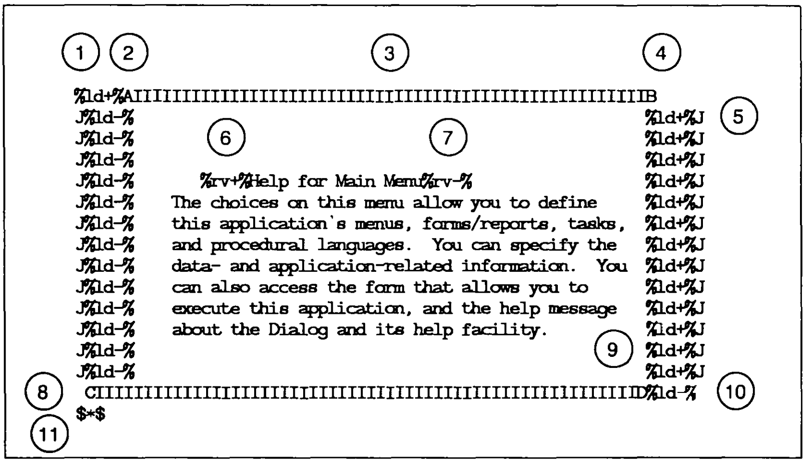
A text file can contain the highlight and line-draw directives shown in Table 4-2. Highlight and line-draw directives begin and end with a percent sign (%). Highlighting is used most often to set off blocks of text. Line-draw directives begin and end with "%ld+" and "%ld-," respectively. Line-draw characters are useful for drawing borders around text areas. Text areas can be highlighted, but line-draw characters cannot share the same space with text.

Table 4-2. Highlight and Line-Draw Directives

Directive	Purpose
%	highlight delimiter that surrounds all highlighting directives
no+	none
ul+	underlining starts
ul-	underlining stops
hi+	half intensity starts
hi-	half intensity stops
rv+	reverse video starts
rv-	reverse video stops
bl+	blinking starts
bl-	blinking stops
cu+	cursor box
no-	return to background highlighting
ld+	line drawing on
ld-	line drawing off
A	upper-left corner
B	upper-right corner
C	lower-left corner
D	lower-right corner
E	left T bar
T	upper T bar
G	right T bar
H	bottom T bar
I	horizontal line
J	vertical line
K	crossing line

Each highlight and line-draw directive must be surrounded by the percent sign (%). The percent sign character is not reserved and therefore can also be used in the message text. Each area of line-draw directives must begin with "%ld+" and end with "%ld-." If you have turned line drawing on, any character that is not a line-draw directive turns off line drawing.

Figure 4-3 shows sample message text with highlight and line-draw directives.



```
%ld+%AIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIB
J%ld-%                                                %ld+%J
J%ld-%                                                %ld+%J
J%ld-%                                                %ld+%J
J%ld-%      %rv+%Help for Main Menu%rv-%              %ld+%J
J%ld-%    The choices on this menu allow you to define  %ld+%J
J%ld-%    this application's menus, forms/reports, tasks, %ld+%J
J%ld-%    and procedural languages.  You can specify the  %ld+%J
J%ld-%    data- and application-related information.  You %ld+%J
J%ld-%    can also access the form that allows you to     %ld+%J
J%ld-%    execute this application, and the help message  %ld+%J
J%ld-%    about the Dialog and its help facility.         %ld+%J
J%ld-%                                                %ld+%J
J%ld-%                                                %ld+%J
 CIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIID%ld-%
$*$
```

Figure 4-3. Text With Highlight and Line-Draw Directives

1) The directive (%ld+%) turns on the line-draw characters that are to surround the message.

2) The directive (A) specifies an upper-left corner line-draw character.

3) The directive (I) specifies a horizontal line-draw character.

4) The directive (B) specifies an upper-right corner line-draw character.

5) The directive (J) specifies a vertical line-draw character.

6) The directive (%rv+%) turns on reverse video highlighting in the message title.

7) The directive (%rv-%) turns off reverse video highlighting in the message title.

8) The directive (C) specifies a lower-left corner line-draw character.

9) The directive (D) specifies a lower-right corner line-draw character.

10) The directive (%ld-%) turns off line drawing.
11) The directive ($*$) signals the end of the message and turns off all highlighting and line-drawing.

Message Text Highlighting

Always specify highlighting directives within the bounds of a message, because they apply only to the message that they surround. Thus, each message should have its own highlighting directives.

Some display terminals require a space before and after a highlighted string. Therefore, if you build an ALLY application for these terminals, do not begin or end highlighting between two characters or at the beginning or end of a line (columns one and eighty on a standard display). If you do, the character at the beginning or the end of the line on the user's display is lost.

With the exception of terminals that require a space around highlighted strings, highlighting directives in the stored message text do not depend on terminal type.

Highlighted Cursor Block

You can use cursor highlighting to highlight a cursor block in a message in applications for terminals that do not require a space to turn highlighting on or off. ALLY includes cursor highlighting so that you can mark a cursor position in help message examples. A cursor block within message text cannot be highlighted in applications for terminals that require a space to turn highlighting on and off.

Background Highlighting

A legend, error message, or help message can have background video highlighting that covers the entire message area from border to border. The Message Builder allows you to specify a combination of highlighting styles for background video when you are loading, merging, or unloading message files. The background highlighting styles are:

- underline
- altered intensity
- reverse video
- blink

You can specify more than one directive for a single text area. Any message can be loaded with a background highlighting style without directives in the text by selecting the appropriate option when you invoke the Message Builder. The background is not highlighted unless you include background highlighting directives in the text or select the corresponding option when you invoke the Message Builder.

Options for the AFILE Message Builder

Table 4-3 describes the six AFILE Message Builder options.

Table 4-3. AFILE Message Builder Options

Option	Action
Create AFILE from text file	Load the text file into a new message AFILE, external to your application.
Merge into existing AFILE	Merge the message text file into an existing library AFILE or an application AFILE. If a message number is already in the AFILE, the new message text replaces the old one.

Option	Action
Unload AFILE to text file	Write the contents of a message AFILE or the messages from a regular application AFILE to an ASCII text file. By default, all messages are unloaded as error messages in the e format.
Unload messages as help	Unload all the messages as help messages in the h format. This option can be used only in combination with the unload option.
Wrap lines in the text file	Insert a continuation character in column eighty of any line that exceeds eighty characters. This option can be used only in combination with the unload option.
Background highlighting styles	Highlight the background of the entire message with any combination of: underline altered intensity reverse video blink.

The Unload Option

The unload option writes the contents of a message AFILE or the messages from a regular application AFILE to an ASCII text file. Unloading has no effect on the AFILE. The messages are unloaded as error messages in the e format, unless you also specify the option to unload messages as help messages.

The unload option is typically used to produce a text file that contains messages, help messages, legends, and embedded strings that were created with the Dialog. Another use is to produce an "unwrapped" version of a file with "wrapped" text. It should only be necessary to unload once. Thereafter, you can more easily maintain text outside of an AFILE, using the Message Builder to reload as necessary.

Background Highlight Options

Specify a background highlighting option for the unload option only when the AFILE was loaded with the same background highlighting options. A warning is displayed when the AFILE was not loaded with the same background.

Any background highlighting option specified for the unload operation is ignored when the AFILE messages do not have background video.

When the background highlighting option specified on unload is different from the background highlighting in the AFILE, the output text will have excessive highlighting directives.

Invoking the Message Builder from the Dialog

Figure 4-4 shows the location of the Dialog forms and subforms that you use to invoke the AFILE Message Builder utility.

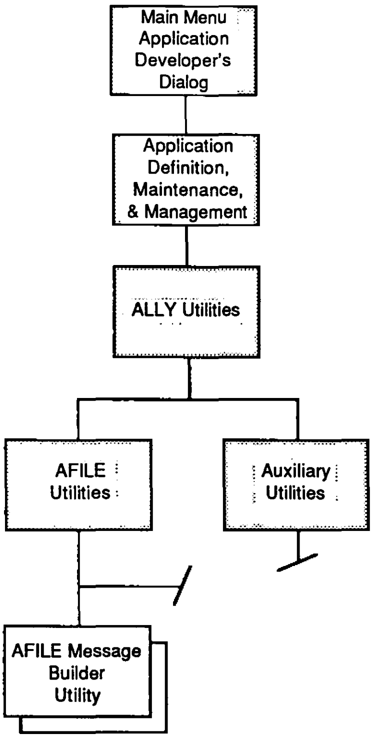
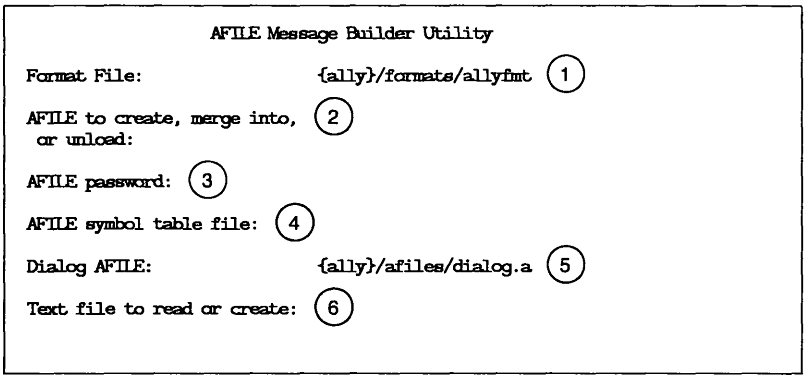

Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
AFILE Message Builder Utility
F002-0581-01

Figure 4-4. Dialog Path to the AFILE Message Builder

Figure 4-5 shows the form that you use to invoke the Message Builder.



```
                     AFILE Message Builder Utility

Format File:                      {ally}/formats/allyfmt  (1)

AFILE to create, merge into,  (2)
 or unload:

AFILE password:  (3)

AFILE symbol table file:  (4)

Dialog AFILE:                     {ally}/afiles/dialog.a  (5)

Text file to read or create:  (6)
```

Figure 4-5. Invoking the AFILE Message Builder

1) The current Format File name is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

The remaining fields require the following information:

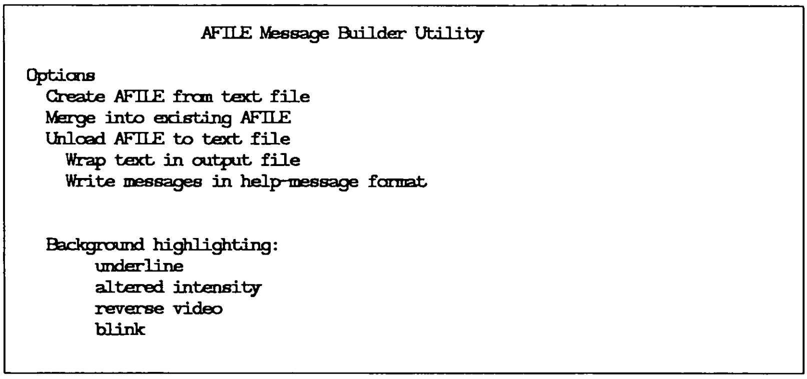
2) The name of the AFILE that is to be created, merged into, or unloaded.

3) The password for the AFILE. Type <Return> if you are not assigning a password to a new AFILE, or if an existing AFILE is not password-protected. Characters that you type in this field are not displayed. By default, an AFILE is not password-protected.

4) The name of the symbol table file, if the AFILE has one. If this AFILE does not have an external symbol table file, type <Return>.

5) The name of the Dialog AFILE that you used to build your application. The current Dialog AFILE is already filled in. Type <Return> to use this AFILE, or edit or delete the displayed name to use a different Dialog AFILE.

6) The name of the file that contains your message list. (If you store legends in a file separate from help messages, you need to invoke the Message Builder from your operating system's command line to include both files at once.) After you complete this form, a subform appears that lists the Message Builder options (Figure 4-6)



```
                    AFILE Message Builder Utility

 Options
   Create AFILE from text file
   Merge into existing AFILE
   Unload AFILE to text file
     Wrap text in output file
     Write messages in help-message format


   Background highlighting:
        underline
        altered intensity
        reverse video
        blink
```

Figure 4-6. Message Builder Options

Type "X<Return>" or "Y<Return>" after any option that you want to select. You must choose one of the options from the first group, that is, you must choose either to create, merge, or unload a file. Each succeeding "X" or "Y" causes the previous one to disappear because you can make only one choice among these three.

After you enter the information required and use the 'exit action' command, the Message Builder executes. The cursor moves to the lower-left corner of the form while the Message Builder runs. After processing is completed, the cursor moves to the menu titled *AFILE Utilities*.

Invoking the Message Builder from the Command Line

The operating-system command file for this utility automatically uses the current Format File.

The command line to invoke the Message Builder from your operating system is:

```
newmsg [AFILE] [options] [text file(s)] [password] [symbol table file] [trunk AFILE]
```

You are required to specify the name of the message AFILE, at least one option, and the name of the text file. The rest of the arguments default to:

- password = none (not password-protected)
- symbol table = none (an internal symbol table)
- trunk AFILE = the Dialog AFILE currently being used

On the command line, you must specify for the arguments:

AFILE	The name of the AFILE to create, merge into, or unload.
options	One of the following three options:

l	load text file(s) to create a new message AFILE
m	merge text file(s) into an existing AFILE
u	unload messages from an AFILE into a text file, defaulting message type to "error."

If desired, any of the options listed below. See the "Options for the AFILE Message Builder" section for details.

b	background highlighting, blink

h	when combined with the "u" option, all messages default to "help" type
i	background highlighting, altered intensity
n	background highlighting, underline
v	background highlighting, reverse video
w	wrap text in the output file, rather than formatting lines to exceed eighty columns. You can use this option only in conjunction with the "u" option to unload an AFILE.

text file(s)	The name of the text file(s) to load, or single text file to create on unload. On some systems, you may need to put the names within double quotes separated by blanks, as "file1 file2 file3." To display syntax information for your system, type the command file name, followed by "help" (e.g., "newmsg help<Return>").
symbol table file	"none" or the name of the external symbol table file for the AFILE.
password	The password for the AFILE. Type "none" if you are not assigning a password to a new AFILE, or if an existing AFILE is not password-protected. By default, an AFILE is not password-protected.
trunk AFILE	When you are creating a message AFILE, the name of the Dialog AFILE that you use to create AFILEs.

Invocation Example

```
newmsg message.a uhv message.txt locked
```

This example unloads a message AFILE (message.a) into a text file named "message.txt." The "u," "h," and "v" options specify that the Message Builder is to:

- unload the message AFILE
- put the messages in the help format (h)
- use reverse video background highlighting

The "message.a" AFILE is protected with the password "locked." Note that because the remaining arguments are omitted, the default values for the symbol table file (none) and trunk AFILE (the current Dialog AFILE) are used.

AFILE Message Builder Error Messages

The AFILE Message Builder produces an error message when it finds:

- that the "c," "m," or "u" option has not been selected to either create, merge, or unload a file
- an invalid password
- a non-numeric character in the error message's number field
- messages with duplicate numbers
- legend messages that are larger than the display area defined for them
- that you are trying to merge text into an AFILE with a lower version number

End of Chapter 4

Chapter 5
The AFILE Script Writer

Introduction

The Script Writer produces text descriptions of ALLY application AFILEs. The input to the Script Writer utility is an AFILE and the output is an ASCII text file that describes the AFILE. There are options to select how much information you want the Script Writer report to contain.

The Script Writer allows you to have a paper backup of your application and aids maintenance and enhancements by providing reference material for your application.

Figure 5-1 illustrates the relationship of the Script Writer utility to its input and the output it produces.

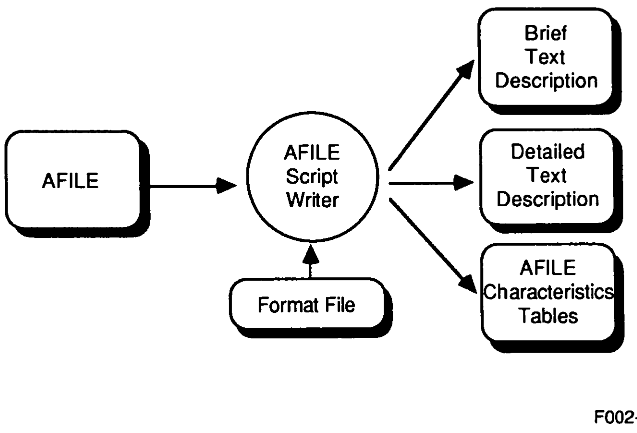


Figure 5-1. The AFILE Script Writer

Script Writer Reports

The Script Writer can produce three types of reports: a brief report, a detailed report, and a list of tables that shows the characteristics of the AFILE's items.

In a Script Writer report, AFILE items are grouped by item type and are listed in alphabetical order. Developer comments are written to the script file as the item's first descriptive entry. Control characters in the AFILE text are written to the output file in the form "\xxx", where "\xxx" is the octal number for the character. Script Writer reports overwrite any existing files with the same name.

The Brief Script Writer Report

The brief Script Writer report shows an AFILE's structure and flow of control. The following items are described by a brief Script Writer report.

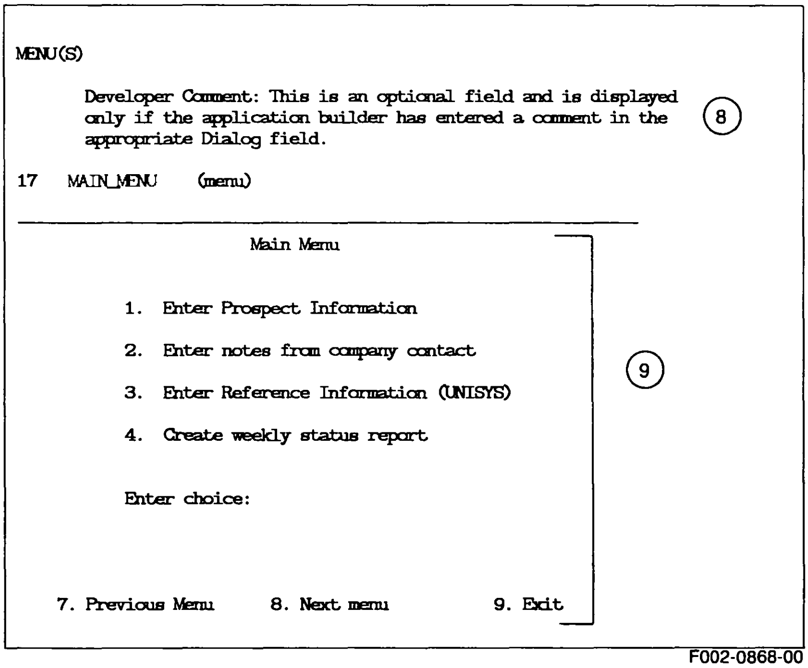
Table 5-1. Brief Report Items

Tasks
Action lists
Menus
Forms/reports
Data Source Definitions
ADL (ALLY Development Language) procedures
Parameter packets

Figure 5-2 shows a sample brief report.

```
          SCRIPT WRITER REPORT OF AFILE: mkt.a  (1)

Script Write created on : 11/18/ 1987 14:56:48  (2)
Format File Used : Version 1.5                (3)
AFILE Used : Version 9                  (4)

TASK(S)

 (5)   (6)     (7)

0    MAIN_TASK    (task)                                      ┐
          Task Action: CALL main_action    (action list)      │
                                                              │
1    main_menu3_task    (task)                                │
          Task Action: CALL main_menu3    (menu)              │
                                                              │
2    main_menu_task    (task)                                 │
          Task Action: CALL MAIN_MENU    (menu)               │  (9)
                                                              │
3    menu2_task    (task)                                     │
          Task Action: CALL main_menu2    (menu)              │
                                                              │
ACTION LIST(S)                                                │
                                                              │
4    contact_notes_al    (action list)                        │
                                                              │
          Action 1: CALL_CMD EXPLODEWDW    (command)          │
          Action 2: CALL contact_notes_form_PKT    (F/R packet)
          Action 3: CALL_CMD RESIZEWDW    (command)           │
                                                              │
                          .                                   │
                          .                                   │
                          .                                   ┘
```

continued



```
MENU(S)

        Developer Comment: This is an optional field and is displayed
        only if the application builder has entered a comment in the      (8)
        appropriate Dialog field.

17   MAIN_MENU    (menu)

_______________________________________________________________
                         Main Menu

            1.  Enter Prospect Information

            2.  Enter notes from company contact                          (9)

            3.  Enter Reference Information (UNISYS)

            4.  Create weekly status report


            Enter choice:



   7. Previous Menu      8. Next menu           9. Exit
```

F002-0868-00

Figure 5-2. Sample Script Writer Brief Report

The Script Writer description lists:

1) The name of the AFILE being described.
2) The date the Script Writer report was produced.
3) The Format File's version number.
4) The AFILE's version number.
5) A unique reference number that can be used to locate and identify each item in the script file.
6) An AFILE item's name.
7) An AFILE item's type (menu, form/report, etc.).
8) Any comments about an item that the developer wrote with the Dialog.

9) A description of each feature of an item. For example, a menu description shows the menu image, and a form/report description lists each group, followed by the structure of the group and its fields.

The Detailed Script Writer Report

An application developer can use the detailed Script Writer report as a debugging aid. A detailed report can describe a single AFILE item, all items of a specific item type, or the entire AFILE. A detailed report gives all information available for the items described.

AFILE items are listed alphabetically unless the entire AFILE is being described. If the report is of the entire AFILE, the items are listed in the order shown in Table 5-2. Thus, form/report fields and groups can be interspersed, but are identified by appropriate labels.

Table 5-2. Detailed Report Items

Tasks	Data Source Definitions
Action Lists	File-dependent information
Menus	Fields
Forms/reports	Keys
Packets	Foreign key links
Groups	ADL procedures
Fields	External programs
Text	Security information*
Conditional next fields	Number formats
Validation	Number character sets
Legends	Date format structure
Conditional LOVs	Embedded Strings
	Parameter packets

* When the security option is specified

The items listed in Table 5-3 are also written to the script file when you specify the option to report global information.

Table 5-3. Global Information Items

Help file information
Error file information
Library AFILE description
Logon set (the access methods used)
Date picture symbols
Character set
Command-to-key assignments
Global variables

The detailed report, shown in Figure 5-3, contains all of the information shown for the brief report and several lines of additional information.

```
          SCRIPT WRITER REPORT OF AFILE: sample.a


Script Write created on : 11/18/ 1987 14:56:48
Format File Used : Version 1.5
AFILE Used : Version 9

                        .
                        .
                        .

   5    SPEC_TOOL_REPORT                              main group
            Form/Report OUTPUT Order:
                TX                (before down text)
                SPEC_TOOL          (data source group)
                    PROJ_NAME          (data source field)
                    PROJ_NUM           (variable field)
                CUR_DATE          (data source field)
                CUR_NUM           (data source field)
            Form/Report OUTPUT Order:
                CUR_DATE          (data source field)
                CUR_NUM           (data source field)
                SPEC_TOOL          (data source group)
                    PROJ_NAME          (data source field)
                    PROJ_NUM           (variable field)
        Group coordinates:  (1,1) to (24,80)
        Page Options Set:
            - Display on first page
        Format Options Set:
            - Indivisible
            - Left justify in window
            - Right justify in window
```

Figure 5-3. Sample Script Writer Detailed Report

Script Writer Directory Listing

At the end of both the brief and detailed Script Writer reports is an alphabetical directory that lists each processed item's name and type. A reference number precedes each directory entry to help you locate the item in the report and the directory. A directory listing is not generated when you specify the option to describe a single item.

Figure 5-4 shows a portion of a sample Script Writer directory output. The reference numbers are listed under the "ID" column heading.

```
DIRECTORY of items

ID      ITEM_NAME
                              ITEM_TYPE
-----------------------------------------------------------------

1     MAIN_MENU.
                               (menu)
0     MAIN_TASK.
                               (task)
          .
          .
          .
7     employee_form.employee
                              (F/R field)
8     employee_form.hire_date
                              (F/R field)
9     employee_form.name
                              (F/R field)
10    employee_form.salary
                              (F/R field)
11    employee_form.ssn
                              (F/R field)
```

Figure 5-4. Sample Script Writer Directory Output

Script Writer Reports of ADL Procedures

The Script Writer prints the text of all ADL procedures in an AFILE, as shown in Figure 5-5.

```
①  ②    ③
123 ADL  mode_adl

 VAR
    mode : NUMBER GLOBAL;           ④

 BEGIN
    mode := 1;
 END;
```

Figure 5-5. Script Writer Description of an ADL Procedure

In Figure 5-5,

1) is the AFILE item's unique reference number
2) is a label that identifies the item as an ADL procedure
3) is the item's name
4) is the body, or text, of the ADL procedure

AFILE Characteristics Tables

When you produce a detailed report, you can also create a file that contains a series of tables showing the characteristics of certain AFILE items. Table 5-4 shows the items for which you can generate characteristics tables. In addition, for forms/reports, and their groups and fields, the tables show before- and after-events.

The name of the file that contains the characteristics tables is created by appending an extension of ".tab" to the file name of the detailed report. Do not specify an extension when you name the file for the detailed report when you produce characteristics tables, because some systems allow only one file name extension.

Table 5-4. Items in Characteristics Tables

Tasks
Menus
Form/report packets
Form/report groups
Form/report text
Form/report fields
Data Source Definitions
Data source fields

The AFILE characteristics tables are particularly useful when you create a Script Writer report of either a single item or of all items of a specific type. If you choose to produce characteristics tables with a detailed report of the entire AFILE, the Script Writer takes a considerable amount of time to process and produces a large output file.

Sample characteristics tables are shown in Example 5-1.

Example 5-1. Tables of AFILE Characteristics

```
          SCRIPT WRITER REPORT OF AFILE: test.a
               Summary Options File


Format File Used : Version 1.5

Task(s) Options Summary:

    Options
   __A___B___C___D___E___F__
   |_*_|___|___|_*_|_*_|___| 0 MAIN_TASK
   |___|___|_*_|___|___|___| 1 TASK_TWO

   A - No toggle in
   B - No toggle out
   C - No key out
   D - No window keys
   E - Beep on startup
   F - Entry point

_________________________________________________________________________
Menu(s) Options Summary:

   Options
   __A___B___C___D___E___F___G___H___I___J___K___L__
   |_*_|___|___|___|___|_*_|___|_*_|___|___|___|___| 4 MAIN_MENU

   A - Numeric menu
   B - Function key menu
   C - Cursor roam menu
   D - Letter menu
   E - Word menu
   F - Roam highlighting
   G - Case sensitive
   H - Minimal match
   I - Jump allowed
   J - Paths allowed
   K - Mixed start on roam
   L - Arrow highlighting
```

continued

```
Form/Report Packet(s) Options Summary:

    Options
    __A___B___C___D___E___F___G___H__
    |_*_|___|___|___|___|_*_|___|___| 7 CREATE_NEW_PKT
    |_*_|___|___|___|___|_*_|___|___| 33 TOOL_PKT

    A - Do not commit on exit
    B - Defer update
    C - Next field generates new record
    D - Previous field generates new record
    E - Exit on any keystroke
    F - Local function action initially active
    G - Commit on queries
    H - Query command not allowed
```

```
    F/R Packet(s) Event Summary:

    Options
    __A___B___C___D___E___F__
    |_*_|___|___|___|___|___|     3    INV_NAME_LIST_RPT_PKT
    |___|___|___|_*_|___|___|     4    INV_STATUS_RPT_PKT
    |___|___|___|___|___|___|     5    LOV_MAINT_PKT
    |___|___|___|___|_*_|___|     6    ORDERS_OUTSTANDING_PKT
    |___|_*_|___|___|___|___|     7    PRINT_NAME_LIST
    |___|_*_|___|___|___|___|     8    PRINT_PROD_STATUS_PKT
    |___|_*_|___|___|___|___|     9    PROD_STATUS_RPT_PKT
    |___|___|___|___|___|___|     10    PUB_INPUT_FORM_PKT

    A - Before-event
    B - After-event
    C - Before-query event
    D - After-query event
    E - Before-commit event
    F - After-commit event
```

continued

```
Form/Report SPEC_TOOL_REPORT:

Form/Report Main Group(s) Options Summary:

    Location Options
    __A__B__C__D__
    |___|___|___|___| 21 SPEC_TOOL_REPORT

    A - Start line relative to group end coordinate
    B - Start column relative to group end coordinate
    C - End line relative to group end coordinate
    D - End column relative to group end coordinate

    Page Options
    __A__B__C__D__F__
    |_*_|___|___|___|___| 21 SPEC_TOOL_REPORT

    A - Display on first page
    B - Display on middle pages
    C - Display on last page
    D - Start new page on first page
    E - Start new page on middle pages
    F - Start new page on last page

    Format options
    __A__B__C__D__F__
    |___|_*_|_*_|_*_|___| 21 SPEC_TOOL_REPORT

    A - Compressible
    B - Indivisible
    C - Left justify in window
    D - Right justify in window
    E - Top justify lines
    F - Bottom justify lines
```

Options for the AFILE Script Writer

The options for Script Writer reports are described in Table 5-5.

Table 5-5. AFILE Script Writer Options

Option	Action
Create a brief Script Writer report	Gives a concise description of items in the AFILE. You cannot specify any other options when you produce a brief report.
Create a detailed Script Writer report	Gives a comprehensive description of items in the AFILE.
Create tables showing the characteristics of the AFILE items	Creates a file that contains tables of the AFILE characteristics, in addition to producing a detailed report.
Describe security in output file	Includes a list of AFILE items that are protected by a password. By default, the Script Writer does not list protected AFILE items. You can use this option only when you create a detailed report.
Describe global information	Includes a description of the global information contained in the first block (512 bytes) of an AFILE and those items that the first block points to. You can use this option only when you create a detailed report.
Describe a single item	Describes the item that you name on the subform that is displayed when you choose this option. This option can be used for any item type listed in Table 5-6.

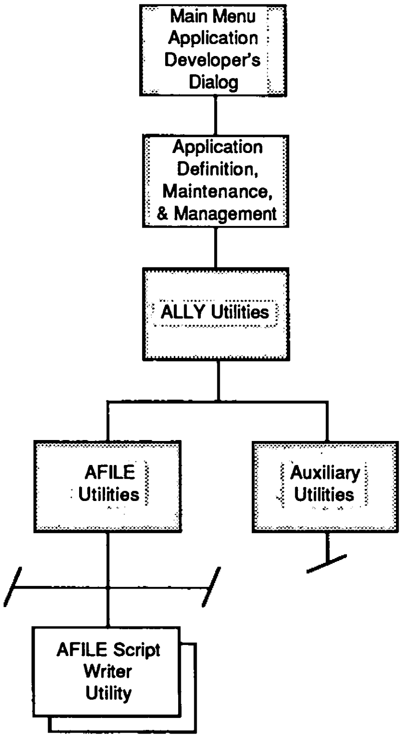
Option	Action
Describe all items of one type	Describes all items of a single type in a detailed report. You name the item type in the subform that is displayed when you choose this option. This option can be used for any item listed in Table 5-6.

Table 5-6. Item Types for Script Writer

TASK
ACTION_LIST
MENU
FORM/REPORT
FORM/REPORT_PKT
DATA_SOURCE
ADL_PROCEDURE
EXTERNAL_LINK
PARAMETER_PKT

Invoking the AFILE Script Writer from the Dialog

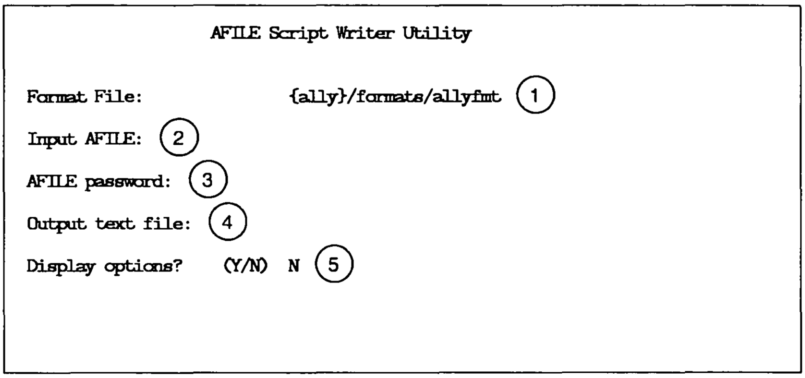
Figure 5-6 shows the location of the Dialog forms and subforms you use to invoke the AFILE Script Writer utility.


Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
AFILE Script Writer Utility

F002-0812-00

Figure 5-6. Dialog Path to the AFILE Script Writer

Figure 5-7 shows the Dialog form that you use to invoke the Script Writer.



```
                    AFILE Script Writer Utility

Format File:                   {ally}/formats/allyfmt  (1)

Input AFILE:  (2)

AFILE password:  (3)

Output text file:  (4)

Display options?    (Y/N)  N  (5)
```

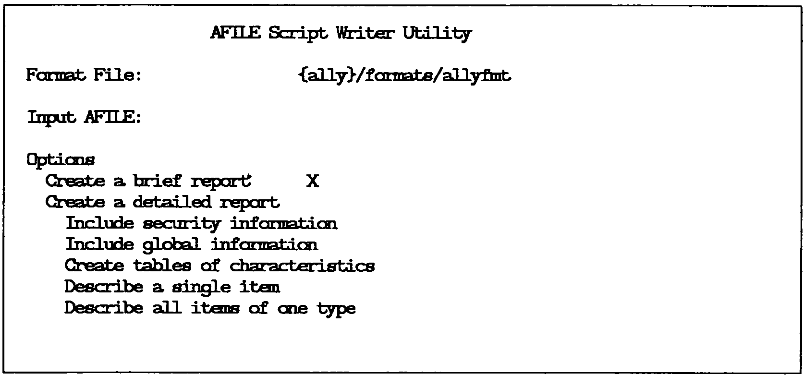
Figure 5-7. Invoking the AFILE Script Writer

1) The name of the current Format File is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

The remaining fields require the following information:

2) The name of the AFILE that you want to describe.
3) The password assigned to the AFILE you want the Script Writer to process or <Return> if the AFILE is not password-protected. Characters that you type in this field are not displayed. By default, the AFILE is not password-protected.
4) The name of the text file that the Script Writer is to produce. Do not specify an extension if you are planning to also create characteristics tables.
5) The default value of "N" is displayed, which bypasses the subform that displays Script Writer options. The Script Writer produces a brief report if you type <Return>.

   When you enter "Y<Return>" or "X<Return>," a subform (Figure 5-8) appears that lists the options for Script Writer reports.



```
                    AFILE Script Writer Utility

Format File:               {ally}/formats/allyfmt

Input AFILE:

Options
  Create a brief report      X
  Create a detailed report
    Include security information
    Include global information
    Create tables of characteristics
    Describe a single item
    Describe all items of one type
```

Figure 5-8. AFILE Script Writer Options

An "X" is displayed in the option field for creating a brief report. If you do not want a brief report, type <Return> to move to the next field, and type "X<Return>" to select the option for creating a detailed report. (The "X" disappears from the field for creating a brief report.) You cannot select any other options when you produce a brief report. If you are producing a detailed report, type "Y<Return>" or "X<Return>" after any other option that you want.

If you choose either the option to describe a single item or the option to describe all items of one type, the cursor moves to a subform on which you name the item or the item type.

After you enter the information required and use the 'exit action' command, the Script Writer executes. The cursor moves to the lower-left corner of the form while the Script Writer runs. After processing is completed, the cursor moves to the menu titled *AFILE Utilities*.

Invoking the AFILE Script Writer from the Command Line

The operating-system command file for this utility automatically uses the current Format File.

The command line to invoke the Script Writer is shown below.

```
ascript [input AFILE] [output text file] [password] [options]
```

The arguments are positional and all are required.

On the command line, you must specify for the arguments:

input AFILE	The name of the AFILE to be described.
output text file	The name of the file to contain the Script Writer report. If this file exists, it is deleted and replaced with a new description file.
password	The password of the AFILE to be described. If the AFILE is not password-protected, specify "none" for this argument.
options	One of the following four options:

b	create a brief Script Writer report. You cannot specify any other options when you produce a brief report.
c	describe all items of a single type (followed by an item type listed in Table 5-6). A detailed report is produced.
d	create a detailed Script Writer report.
f	describe only one item from an AFILE (followed by the name of the item exactly as it appears in the application). This option can be used for any item type listed in Table 5-6.

If desired, any of the options listed below. See the "Options for the AFILE Script Writer" section for details.

g	describe the global items contained in the first block of the AFILE in a detailed report. Use this option in conjunction with the "d" option.
s	include security items in a detailed report. Use this option in conjunction with the "d" option.
t	create a file that contains the summary tables of the AFILE characteristics, in addition to producing a detailed report. The file name is whatever you specified as the name of the detailed report. The file extension is ".tab". Because some systems allow only one file name extension, do not specify an extension in the output file argument when you plan to use this option.

Invocation Example

```
ascript sales.a salestxt secure dgst
```

This example produces a detailed Script Writer report (salestxt) on the entire AFILE named "sales.a," which is protected with the password "secure." The "d," "g," "s," and "t" options:

- create a detailed report
- describe global items
- describe security items
- create a file named "salestxt.tab" that contains summary tables of the AFILE characteristics.

AFILE Script Writer Error Messages

The Script Writer produces an error message if you:

- do not specify an input file
- do not specify any option
- specify more than one of the following options: "b," "c," "d," or "f"
- have an invalid item type for the "c" option
- have an invalid symbol table
- have an unusable Format File
- have a file name conflict (e.g., the output and input files have the same name)
- have an output file that cannot be opened
- have an invalid password
- specify a file name extension for the output file when you also specify the option to create characteristics tables

If you get an error message about an "ALLY internal error" detected, there is something wrong with the AFILE you are using as input to the Script Writer. Contact your system manager or ALLY support representative for help.

End of Chapter 5

Chapter 6
The AFILE Migrator

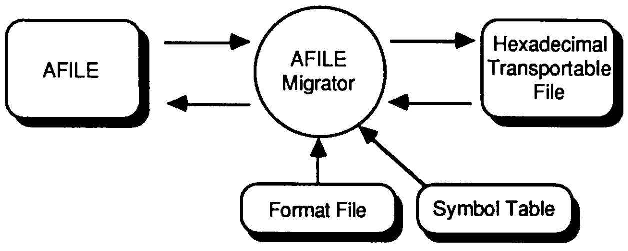
Introduction

The AFILE Migrator utility allows you to transport a copy of your AFILE to a different computer or operating system.

The AFILE Migrator performs two operations:

Text writing	Writes a hexadecimal text file from an AFILE on the source computer system. The AFILE Migrator translates (unloads) a copy of an AFILE on the source computer system to a text file without computer dependencies. The text output file that the AFILE Migrator produces contains hexadecimal codes and is useful only for transporting; that is, it is not readable ASCII output.
Text reading	Reads a hexadecimal text file to reconstruct an AFILE on the target computer system. After the hexadecimal output file is transported to another computer, you use the AFILE Migrator on the target computer system to reconstruct from the transported file an AFILE with computer dependencies appropriate for the target computer system.

The AFILE Migrator's basic operation is shown in Figure 6-1.

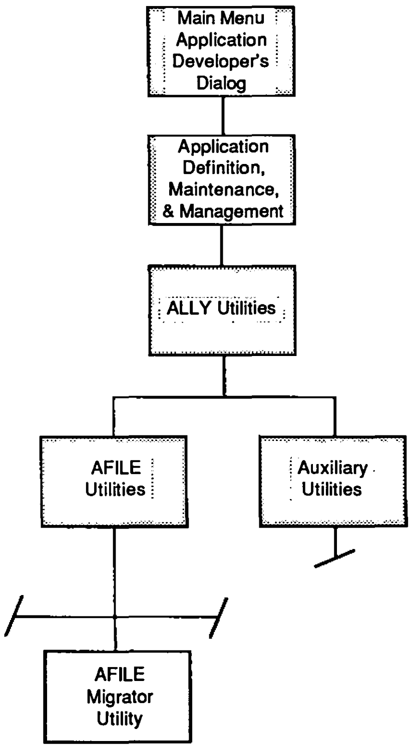


F002-0819-00

Figure 6-1. The AFILE Migrator

Invoking the AFILE Migrator from the Dialog

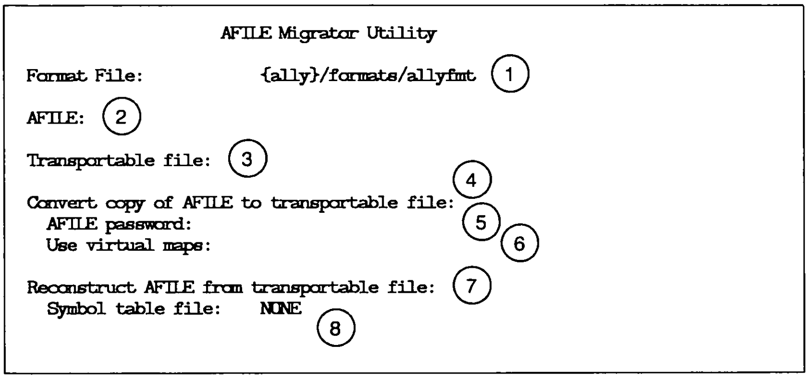
Figure 6-2 shows the location of the Dialog forms and subforms that you use to invoke the AFILE Migrator utility.


Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
AFILE Migrator Utility

F002-0582-01

Figure 6-2. Dialog Path to the AFILE Migrator

Figure 6-3 shows the form that you use to convert a copy of an AFILE to a transportable file or to reconstruct an AFILE from a transported file.



```
                    AFILE Migrator Utility

Format File:             {ally}/formats/allyfmt  (1)

AFILE: (2)

Transportable file: (3)
                                                  (4)
Convert copy of AFILE to transportable file:
  AFILE password:                                 (5)
  Use virtual maps:                                  (6)

Reconstruct AFILE from transportable file:        (7)
  Symbol table file:     NONE
                               (8)
```

Figure 6-3. Invoking the AFILE Migrator

1) The name of the current Format File is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

Enter the following information:

2) The name of the AFILE that you want to convert (unload) or reconstruct (load).

3) The name of the transportable file that you want to create or from which the AFILE is to be reconstructed.

If you want to convert a copy of an AFILE to a transportable file, enter the following information:

4) Type "Y<Return>" or "X<Return>" in the "Convert copy of AFILE to transportable file" field.

5) The password for the AFILE or <Return> if the AFILE is not password-protected.

   Characters that you type into this field are not displayed. By default, an AFILE is not password-protected.

6) Type "Y<Return>" or "X<Return>" if you want the AFILE Migrator to use virtual maps. You need this option when the machine has a limited amount of memory.

Type <Return> if you do not want the AFILE Migrator to use virtual maps.

If you want to reconstruct an AFILE from a transportable file, enter the following information:

7) Type "Y<Return>" or "X<Return>" in the "Reconstruct AFILE from transportable file" field.

8) The name of the external symbol table file for the AFILE or <Return> if the AFILE does not have an external symbol table file.

After you enter the information required and use the 'exit action' command, the AFILE Migrator executes. The cursor moves to the lower-left corner of the form while the AFILE Migrator runs. After processing is completed, the cursor moves to the menu titled *AFILE Utilities*.

Invoking the AFILE Migrator from the Command Line

The operating-system command file for this utility automatically uses the current Format File.

Converting an AFILE to a Transportable File

The command line to invoke the AFILE Migrator to convert a copy of an AFILE to a transportable file is:

amigrate *[input file] [output file] [password]* [**w**] *[v]*

All arguments except the the last one are required. On the command line you must specify for the arguments:

input file	The name of the AFILE that is to be converted (unloaded) to a transportable file.
output file	The name of the transportable file that the AFILE Migrator is to write.
password	The AFILE's password, or "none" if the AFILE is not password-protected. By default, an AFILE is not password-protected.
w	To write a transportable file, that is, to convert an AFILE to a transportable file.

The following is optional:

v	If you want the AFILE Migrator to use virtual maps. You should use this option when your machine has a limited amount of memory.

Invocation Example: From an AFILE to a Transportable File

```
amigrate employee.a employee.t personnel wv
```

This example converts a copy of an AFILE named "employee.a," to a transportable file (employee.t). The "employee.a" AFILE is protected by the password "personnel." The "w" indicates that the AFILE Migrator is to unload "employee.a" and the "v" option specifies that virtual maps are to be used to preserve memory space.

Reconstructing an AFILE

The command line to invoke the AFILE Migrator to reconstruct an AFILE is:

```
amigrate [input file] [output file] [symbol table file] [r]
```

All of the arguments are required. On the command line you must specify for the arguments:

input file	The name of a transportable file that the AFILE Migrator has previously created.
output file	The name of the AFILE that the AFILE Migrator is to reconstruct.
symbol table file	The name of the external symbol table file to create for the AFILE or "none" if you do not want an external symbol table.
r	To reconstruct an AFILE from a transported file.

Invocation Example: Reconstructing an AFILE

```
amigrate employee.t employee.a symtable r
```

This example reconstructs an AFILE (employee.a) from a transported file (employee.t). The "employee.a" AFILE is to have an external symbol table file named "symtable." The "r" indicates that the AFILE Migrator is to reconstruct an AFILE from "employee.t".

AFILE Migrator Error Messages

The AFILE Migrator produces an error message if it finds:

- an invalid AFILE item type
- an incorrect password

End of Chapter 6

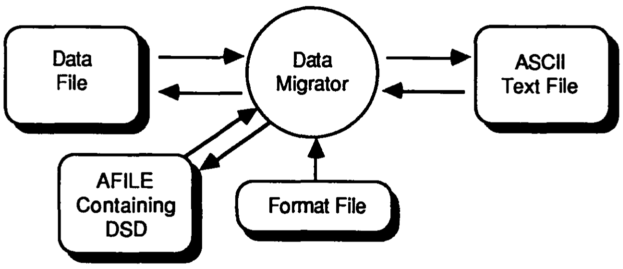
Chapter 7
The Data Migrator

Introduction

The Data Migrator utility (Figure 7-1) allows you to transport a copy of your data to a different access method, or, to a different computer or operating system.

The Data Migrator performs these operations:

Text writing	Describes an application's data in a text file that can be transported to another system.
Text reading	Reconstructs the data from a transported text description file, and puts the data into a dataset, file, or table in the database.



F002-0267-02

Figure 7-1. The Data Migrator Utility

The Text Writing Operation

The Data Migrator uses a Base Data Source Definition (Base DSD) in an AFILE to produce a text file that describes the data in an ALLY-supported file, table, or dataset. Figure 7-2 shows an example of the text data description file that the Data Migrator produces. The parts of a Data Migrator writing operation are labeled and explained in the paragraphs that follow the example.

```
        Data Referenced by Base Definition: employee  ①

There are 4 fields in each record
CHAR (25) DATE CHAR (25) NUMBER (FIX,7,2)  ②

The following is a script write of the actual data :
(8):Jane Doe (19):01/05/1978 00:00:00 (9):177439456 (5):24000  ③

(8):John Doe (19):06/08/1984 00:00:00 (9):077349834 (5):18000

(12):Joe Johnson (19):04/01/1983 00:00:00 (9):123456789 (5):21000  ④
```

Figure 7-2. Sample Text Description of Data from a DSD

The Data Migrator description contains:

1) A header indicating the source of the data (the Base DSD name).
2) The number of fields in a record and the attributes of each field.
3) A number within parentheses that precedes each field value and tells the Data Migrator the number of characters to read in at a time.
4) The text description of the data.

The format of this text file is independent of the access method used to store the data. Each record from the dataset, file, or table is printed as a continuous stream of characters with a delimiter between the fields, spanning multiple lines as needed. A line containing a newline character is used to separate the records. All of the text strings and delimiters used for output are specified in the Format File. Any control sequence that is found in character data is represented by its octal sequence, preceded by a backslash (\).

The Text Reading Operation

The Data Migrator allows you to use the information from the Base DSD description to insert or append data records to a dataset, file, or table. The Data Migrator processes the text file to:

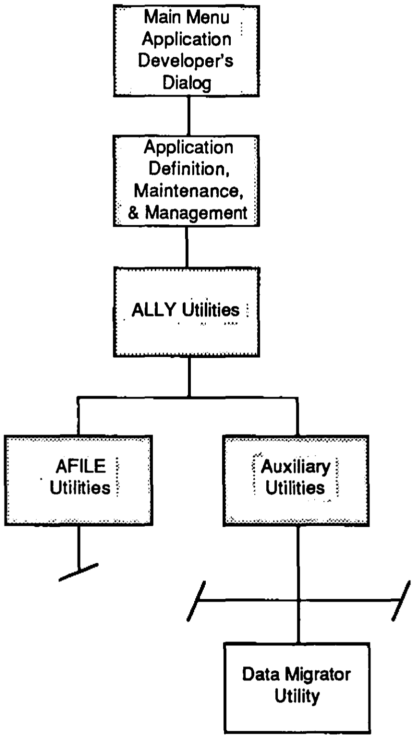
- create a new data file if it does not already exist
- append new records to an existing data file, if it is fixed sequential in the same record format
- insert new records to an existing data file, if the access method supports data insertion

The Data Migrator checks that the number of fields and their data types match. But because it does not check any other access-method dependencies, the Data Migrator cannot tell you whether the output dataset, file, or table is valid.

For a fixed sequential data file, the output file has the name that is specified in the access-method dependent structures of the Base DSD. If the record format differs from an existing data file, the resultant output file may be unusable.

Invoking the Data Migrator from the Dialog

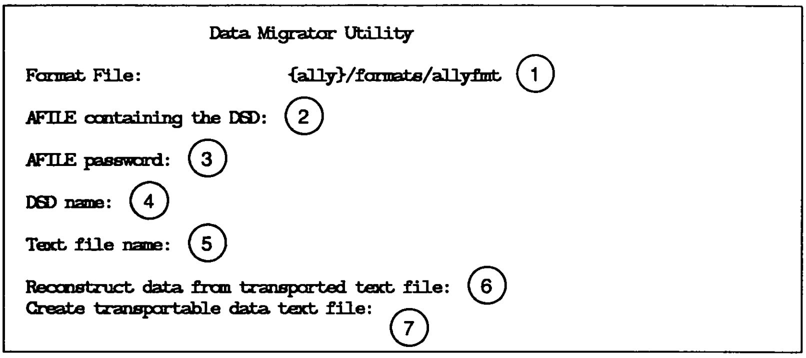
Figure 7-3 shows the location of the Dialog forms and subforms that you use to invoke the Data Migrator utility.


Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
Data Migrator Utility

F002-0815-00

Figure 7-3. Dialog Path to the Data Migrator Utility

Figure 7-4 shows the Dialog form that you use to invoke the Data Migrator.



```
                    Data Migrator Utility

Format File:                 {ally}/formats/allyfmt (1)

AFILE containing the DSD: (2)

AFILE password: (3)

DSD name: (4)

Text file name: (5)

Reconstruct data from transported text file: (6)
Create transportable data text file:
                                          (7)
```

Figure 7-4. Invoking the Data Migrator

1) The name of the current Format File is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

Enter the following information in the remaining fields:

2) The name of the AFILE that contains the DSD for the data you are migrating.
3) The password for the AFILE that contains the DSD. Type <Return> if the AFILE is not password-protected.

   Characters that you type into this field are not displayed. By default, an AFILE is not password-protected.
4) The name of the DSD for the data you are migrating.
5) The name of the text file that the Data Migrator is to create or reconstruct.
6) Type "Y<Return>" or "X<Return>" if you want to reconstruct the AFILE's data from the transported text file. The data is appended or inserted into a dataset, file, or access method table.
7) Type "Y<Return>" or "X<Return>" if you want to create a text file describing the data that can be transported to another system. The data description is written to a text file from a data file that the Base DSD references.

After you enter the information required and use the 'exit action' command, the Data Migrator executes. The cursor moves to the lower-left corner of the form while the Data Migrator runs. After processing is completed, the cursor moves to the menu titled *Auxiliary Utilities*.

Invoking the Data Migrator from the Command Line

The operating-system command file for this utility automatically uses the current Format File.

The command line to invoke the Data Migrator utility is:

```
dmigrate [AFILE] [password] [DSD name] [text file]
[r or w]
```

All of the arguments are required. On the command line you must specify for the arguments:

AFILE	The name of the AFILE that contains the Base DSD.
password	The AFILE's password, or "none" if the AFILE is not password-protected. By default, an AFILE is not password-protected.
DSD name	The name of the Base DSD that references an external data file.
text file	The name of the text description file that the Data Migrator is to either produce from the data, or the text file that is to be written to a data file.
r	Reconstruct data from a transported text description file. The data is created, appended, or inserted into a dataset, file, or access method table that can be referenced by a Base DSD.

w	To create (write) a text file describing the data that can be transported to another system.

Invocation Example

```
dmigrate employee.a personnel datafile_fx employee.txt w
```

This example writes a description of the data in a file referenced by the "datafile_fx" Base DSD. The data description is written to a text description file (employee.txt) that can be transported to another system. The AFILE named "employee.a" is protected with the password "personnel" and contains the "datafile_fx" Base DSD. The "w" indicates that the Data Migrator is to write a text description file of the data that "datafile_fx" references.

Data Migrator Error Messages

The Data Migrator produces an error message if it finds:

- an incorrect number of command line arguments.
- no direction for the operation specified (i.e., read or write).
- field attributes in the text file that do not match those defined by the given Base DSD.
- an error in reading a value, premature end of file, or any other damage to the text file.
- the existence of an output text file of the same name. This is a warning to prevent the text-writing operation from overwriting a file with the same name as the one specified for the output text file. Note that the text reading operation does not perform this check.

End of Chapter 7

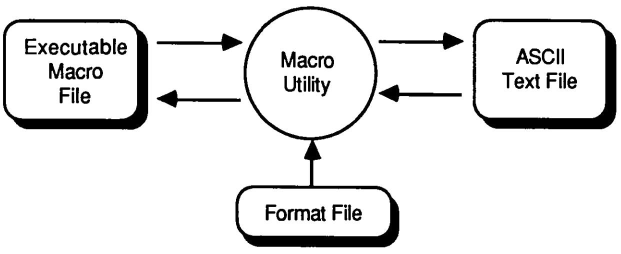
Chapter 8
The Macro Utility

Introduction

When you are working in an ALLY application, you may often use the same series of keystrokes (text or commands). To avoid having to repeat keystrokes, you can store them in an ALLY macro that you can invoke as a group when you need them again. You can create macro files that contain ALLY commands and text while you are working within any ALLY application, including the Dialog. You can then execute all of the keystrokes with a single command.

For example, suppose you frequently repeat several lines of text and commands in the course of a particular project. If you define a macro that contains these commands and text, whenever you invoke the macro, the commands execute and the text is displayed on your terminal.

The Macro Utility (Figure 8-1) allows you to transport macro files to different operating systems that run ALLY. The Macro Utility also allows you to create an ASCII text version of a macro and to edit the ASCII file to change macros without having to redefine them. Conversely, the Macro Utility can use these text files to create executable macro files.

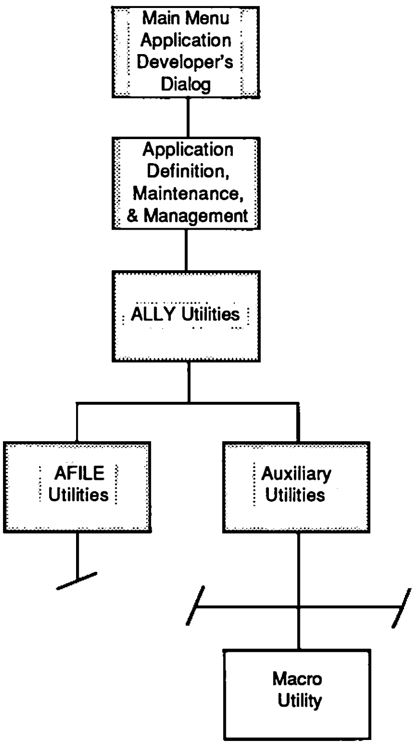


F002-0580-00

Figure 8-1. The Macro Utility

Invoking the Macro Utility from the Dialog

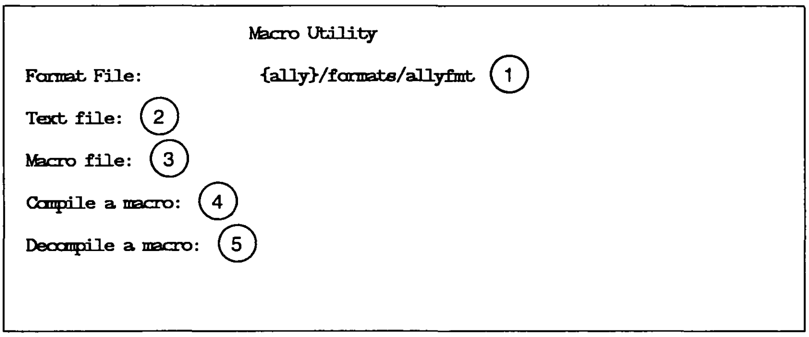
Figure 8-2 shows the location of the Dialog forms and subforms you use to invoke the Macro Utility.


Main Menu Application Developer's Dialog
Application Definition, Maintenance, & Management
ALLY Utilities
AFILE Utilities
Auxiliary Utilities
Macro Utility

F002-0816-00

Figure 8-2. Dialog Path to the Macro Utility

Figure 8-3 shows the form for invoking the Macro Utility.



```
                    Macro Utility
Format File:         {ally}/formats/allyfmt  1
Text file:  2
Macro file:  3
Compile a macro:  4
Decompile a macro:  5
```

Figure 8-3. Invoking the Macro Utility

1) The name of the current Format File is filled in for you. Type <Return> to use this file, or, edit or delete the displayed name to use a different Format File.

Enter the following information in the remaining fields:

2) The name of the text file to compile into a macro or the file to create from decompiling a macro.

   The text file must have been decompiled with the Macro utility in order to be recompiled.

3) The name of the file of macro commands.

4) Enter "Y<Return>" or "X<Return>" if you want to compile a macro into a text file.

5) Enter "Y<Return>" or "X<Return>" if you want to decompile a text file into a macro.

After you enter the information required and use the 'exit action' command, the Macro Utility executes. The cursor moves to the lower-left corner of the form while the Macro Utility runs. After processing is completed, the cursor moves to the menu titled *Auxiliary Utilities*.

Invoking the Macro Utility from the Command Line

The operating-system command file for this utility automatically uses the current Format File.

The command line to invoke the Macro Utility is:

```
mmigrate [input file] [output file] [c or d]
```

All of the arguments are required.

To compile a macro, specify:

input file	The name of the text file previously produced by decompiling a macro.
output file	The name of the executable macro file to produce.
c	To compile a text file into a macro file.

To decompile a macro, specify:

input file	The name of the file of macro commands to decompile.
output file	The name of the text file to produce.
d	To decompile a file of macro commands into a text file.

Invocation Example

```
mmigrate macfile.me textfile.mt d
```

This example decompiles a macro (macfile.me) into a text file named "textfile.mt." We have used an ".me" extension to indicate a "macro executable" file, and the ".mt" extension to indicate a "macro text" file. The "d" indicates that the Macro Utility is to decompile the input file.

Macro Utility Error Messages

The Macro Utility produces an error message if it finds:

- an incorrect number of command line arguments
- an invalid option on the command line
- an empty input file

Editing a Decompiled Macro

If you have stored many commands or text in a macro file and want to change the function of the macro, you may find it easier to edit the macro, rather than rebuild it.

To edit a macro, you first need to decompile the macro into an ASCII text file. You can then edit the text file to change the way the macro functions. Use the Macro Utility again to compile the text file back into a macro.

Before you edit a macro, you need to understand the following characteristics of a macro text file:

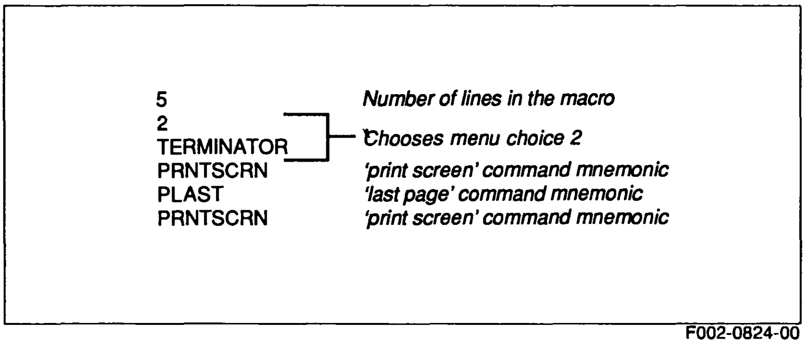
- The first line of a macro contains a number that represents the number of lines in the macro. When a file contains several macros, the first line containing the line count marks the end of one macro and the beginning of the next.

Therefore, if you change the length of a macro by adding or deleting lines, you must also update the number in the first line.

- Each ALLY command in the macro is represented by the command's mnemonic, and is listed on a separate line. Command mnemonics are listed in Appendix B.
- Each letter of a character string stored in the macro is listed on a separate line. For example, if you stored the 'find' command and the text string "the word," the character string is stored in the macro as follows:

```
t
h
e
            This space represents the space
            that was entered between "the" and "word"
w
o
r
d
```

Figure 8-4 shows a sample ASCII text file produced by decompiling a macro with the Macro Utility.



```
5                    Number of lines in the macro
2
TERMINATOR      ]--  Chooses menu choice 2
PRNTSCRN             'print screen' command mnemonic
PLAST                'last page' command mnemonic
PRNTSCRN             'print screen' command mnemonic
```

F002-0824-00

Figure 8-4. Sample Decompiled Macro File

When the macro shown above is compiled and executed, the following series of events take place:

1) Menu choice 2 is selected. (The cursor moves to the chosen form/report.)
2) The 'print screen' command is invoked and prints the first page of the form/report.
3) The 'last page' command is invoked and moves the cursor to the last page of the form/report.
4) The 'print screen' command is invoked and prints the last page of the form/report.

If you want to add more commands to your macro file, enter each command mnemonic on a separate line.

If you want to add a text string to your macro, enter each character in the string on a separate line. If there are spaces between words in the text string, separate the words with a blank line by typing <Return>.

You need to keep track of the number of lines you add to or delete from your macro file and update the number in the first line of the macro file (representing the number of lines in the macro) to reflect the changes you have made to the file.

When you have finished editing the ASCII text file, you can use the Macro Utility to compile the file into an executable macro that contains the changes you have made.

End of Chapter 8

Appendix A
Utility Command Lines

This appendix contains a summary of the command lines that you use to invoke the ALLY utilities. The command lines are listed in alphabetical order.

The operating-system command file for a utility automatically uses the current Format File. You can use a different Format File by renaming files. First, preserve a valid Format File by copying the original Format File to another file before renaming it. Then rename the Format File that you want to use with the Format File name that your system is set up to use.

Some command lines do not require that you specify every argument. However, because the arguments are positional, you must provide all of the arguments up to the one you want. This means that if you want only the last argument, you must also give a value for each argument up to the last one.

If you need to provide the password or the symbol table file parameter, and the AFILE does not have a password or an external symbol table file, specify "none." The default value for the password and symbol table arguments is "none," which means that the AFILE is not password-protected and has an internal symbol table, respectively.

The way you specify the arguments can be operating-system dependent. You can enter the command followed by "help" (e.g., "acompact help<Return>") to display syntax information for your system.

The AFILE Compactor

acompact *[input AFILE] [output AFILE] [symbol table file] [password] [options]*

Only the first two arguments are required. If you specify "none" for the symbol table file argument when the input AFILE has an external symbol table, the Compactor restores an internal symbol table to the AFILE.

The Compactor options are:

d	delete disjoint items.
s	suppress statistics.
v	use virtual maps.

The AFILE Merger

amerge *[master AFILE] [master AFILE password] [second AFILE] [second AFILE password] [output AFILE] [symbol table file] [output AFILE password] [options]*

You must specify a minimum of the first three arguments if you are merging the second AFILE into the master AFILE and the second AFILE does not have a password. If you want to create a new merged AFILE, you must specify at least the first five arguments.

The Merger options are:

c	suppress compaction of new AFILE.
d	delete disjoint items.
e	suppress entry point merge.
g	suppress global variable and list merge.
m	resolve conflicts with item from master AFILE.
s	suppress statistics from the AFILE Compactor.
v	use virtual maps.

The AFILE Message Builder

newmsg *[AFILE] [options] [text file(s)] [password] [symbol table file] [trunk AFILE]*

You are required to specify the name of the message AFILE, at least one option ("l", "m", or "u"), and the name of the text file. The rest of the arguments default to:

- password = none (not password-protected)
- symbol table file = none (internal symbol table)
- trunk AFILE = the installed Dialog AFILE

The Message Builder options are:

l	load text file(s) to create a new message AFILE.
m	merge text file(s) into an existing AFILE.
u	unload messages from an AFILE into a text file, making "error" the default message type.
b	background highlighting, blinking.
h	when combined with the "u" option, all messages default to "help" type.
i	background highlighting, altered intensity.
n	background highlighting, underlining.
v	background highlighting, reverse video.
w	wrap text in output file, rather than fully formatting lines and exceeding eighty columns. You can use this option only when you also specify the "u" option to unload an AFILE.

The AFILE Migrator

amigrate *[input file] [output file] [symbol table* or *pass-word] [***r** or **w***] [***v***]*

All of the arguments except the final one are required. Specify one of the following:

r	to reconstruct an AFILE from a transportable file; the preceding argument is the name of the AFILE's external symbol table file, or "none."

When you are reconstructing an AFILE, you can specify the following option as the final argument:

v	use virtual maps. It is useful when your machine has limited memory.

w	to write (create) a transportable file from an AFILE; the preceding argument is the AFILE's password or "none."

The AFILE Script Writer

ascript *[input AFILE] [output text file] [password] [options]*

All of the arguments are required, because you must specify at least one option.

The Script Writer options are:

b	create a brief Script Writer report.

c	describe all AFILE items of one type (category), followed by the name of the item type. A detailed report is produced.

d	create a detailed Script Writer report.

f describe only one item of an AFILE (followed by the name of the item). A detailed report is produced.

g describe the global items contained in the first block of the AFILE. A detailed report is produced.

s include security items in a detailed report.

t create a file that contains tables of the AFILE characteristics, in addition to producing a detailed report.

The Data Migrator

dmigrate *[AFILE] [password] [DSD name] [text file]*
*[***r** or **w***]*

All of the arguments are required. The final argument is one of the following:

r to read a text file, and to create or insert a data area that can be referenced by a Base DSD (named in the "DSD name" argument).

w to write (create) a text file from a data file referenced by a Base DSD (named in the "DSD name" argument).

The Macro Utility

mmigrate *[input file] [output file] [***c** or **d***]*

All of the arguments are required. The final argument is one of the following:

c to compile a text file into a file of executable macro commands.

d to decompile a file of executable macro commands into a text file.

End of Appendix A

Appendix B
ALLY Command Mnemonics

Mnemonic	Command
abortaction	Abort action
abortappl	Abort application
aborttask	Abort task
addnl	Add new line
bdelete	Back delete
bol	Beginning of line
bottom	Bottom
box	Box
budmode	Browse/update/delete mode
clrcasesens	Clear case sensitive
clrdrawmode	Clear draw mode
clrovertype	Clear overtype
clrpwrtype	Clear powertype
commit	Commit
compresswdw	Compress window
cpfrombuf	Copy from buffer
cptobuf	Copy to buffer
ctrlchar	Enter control character
definewdw	Define window
defmacro	Define macro
delbol	Delete to beginning of line
deleol	Delete to end of line
delline	Delete line
delrec	Delete current record
deltomark	Delete to mark
delword	Delete word
down	Down
downpage	Down page
dupdate	Deferred update

Mnemonic	Command
eol	End of line
exemac	Execute macro
exemacf	Execute macro from file
exitaction	Exit action
exitappl	Exit application
exittask	Exit task
expandwdw	Expand window
explodewdw	Explode window
fdelete	Forward delete
fhome	First field
find	Find
findanddel	Find and delete
finsnext	Insert first record in next group
flast	Last field
flistval	Move to list of values
fnext	Next field
fpickval	Pick from list of values
fprev	Previous field
frfunction	Invoke local function
glblreplace	Global replace
hightomark	Highlight to mark
hightypeset	Set highlight type
home	Home
homemch	Home area
ignore	Ignore
insafter	Insert record after
insbefore	Insert record before
insertline	Insert line
jumptomark	Jump to mark
khelp	Help
kmpprint	Print menu

Mnemonic	Command
ldtomark	Line draw to mark
left	Left
loadmacros	Load macros
macfmfile	Macro from file
mactofile	Macro to file
mark	Set mark
menu	Function key choice
movewdw	Move window
nextline	Next line
nextmch	Next area
nextword	Next word
overlayblk	Overlay block
pall	Print all
phome	First page
pickfield	Copy to field-buffer
picktask	Pick task
plast	Last page
pnext	Next page
ppage	Print page
pprev	Previous page
prest	Print rest
prevmch	Previous area
prevmenu	Previous menu
prntvnum	Print version number
prevword	Previous word
prhome	First display area
prlast	Last display area
prnext	Next display area
prntscrn	Print screen
prompt	Prompt line
prprev	Previous display area
putfield	Copy from field-buffer

Mnemonic	Command
qbe	Query by example
query	Execute query
qwhere	Query by where clause
readfile	Read from file
redraw	Redraw
refresh	Refresh
removeblk	Remove block
replace	Replace
resizewdw	Resize window
rghome	First group
rglast	Last group
rgnext	Next group
rgprev	Previous group
rhome	First record
right	Right
rlast	Last record
rnext	Next record
roamfirst	First area
roamlast	Last area
rollback	Rollback
rprev	Previous record
save	Save
savemacros	Save macros
scrollwdw	Scroll window
select	Choose from roam area
setcasesens	Set case sensitive
setdelaycnt	Pause
setdrawmode	Set draw mode
setovertype	Set overtype
setpwrtype	Set powertype
setrptcnt	Set repeat count
shell	Go to OS command line processor

Mnemonic	**Command**
task	Start task
terminator	Choose from prompt line
togcasesens	Toggle case sensitive
togdrawmode	Toggle draw mode
toggletask	Toggle task
togovertype	Toggle overtype
togpwrtype	Toggle powertype
top	Top
topmenu	First menu
turtleclear	Clear turtle
turtlehl	Highlight with turtle
turtleld	Line draw with turtle
uldtomark	Erase line draw
uldturtle	Erase line draw with turtle
unbox	Unbox
undelline	Undelete line
undelrec	Undelete record
undelword	Undelete word
up	Up
uppage	Up page
windone	Window-action
windown	Window down
winleft	Window left
winright	Window right
winup	Window up
writefile	Write to file

End of Appendix B

Appendix C ASCII Character Codes

CTRL	Character	Binary Bit 7 to Bit 0	Octal	Decimal	Hexidecimal	Character	Binary Bit 7 to Bit 0	Octal	Decimal	Hexidecimal
@	NUL	00000000	000	000	00	@	01000000	100	064	40
A	SOH	00000001	001	001	01	A	01000001	101	065	41
B	STX	00000010	002	002	02	B	01000010	102	066	42
C	ETX	00000011	003	003	03	C	01000011	103	067	43
D	EOT	00000100	004	004	04	D	01000100	104	068	44
E	ENQ	00000101	005	005	05	E	01000101	105	069	45
F	ACK	00000110	006	006	06	F	01000110	106	070	46
G	BEL	00000111	007	007	07	G	01000111	107	071	47
H	BS	00001000	010	008	08	H	01001000	110	072	48
I	HT	00001001	011	009	09	I	01001001	111	073	49
J	LF	00001010	012	010	0A	J	01001010	112	074	4A
K	VT	00001011	013	011	0B	K	01001011	113	075	4B
L	FF	00001100	014	012	0C	L	01001100	114	076	4C
M	CR	00001101	015	013	0D	M	01001101	115	077	4D
N	SO	00001110	016	014	0E	N	01001110	116	078	4E
O	SI	00001111	017	015	0F	O	01001111	117	079	4F
P	DLE	00010000	020	016	10	P	01010000	120	080	50
Q	DC1	00010001	021	017	11	Q	01010001	121	081	51
R	DC2	00010010	022	018	12	R	01010010	122	082	52
S	DC3	00010011	023	019	13	S	01010011	123	083	53
T	DC4	00010100	024	020	14	T	01010100	124	084	54
U	NAK	00010101	025	021	15	U	01010101	125	085	55
V	SYN	00010110	026	022	16	V	01010110	126	086	56
W	ETB	00010111	027	023	17	W	01010111	127	087	57
X	CAN	00011000	030	024	18	X	01011000	130	088	58
Y	EM	00011001	031	025	19	Y	01011001	131	089	59
Z	SUB	00011010	032	026	1A	Z	01011010	132	090	5A
[	ESC	00011011	033	027	1B	[	01011011	133	091	5B
\	FS	00011100	034	028	1C	\	01011100	134	092	5C
]	GS	00011101	035	029	1D	]	01011101	135	093	5D
^	RS	00011110	036	030	1E	^	01011110	136	094	5E
_	US	00011111	037	031	1F	_	01011111	137	095	5F
	SP	00100000	040	032	20	‘	01100000	140	096	60
	!	00100001	041	033	21	a	01100001	141	097	61
	"	00100010	042	034	22	b	01100010	142	098	62
	#	00100011	043	035	23	c	01100011	143	099	63
	$	00100100	044	036	24	d	01100100	144	100	64
	%	00100101	045	037	25	e	01100101	145	101	65
	&	00100110	046	038	26	f	01100110	146	102	66
	’	00100111	047	039	27	g	01100111	147	103	67
	(	00101000	050	040	28	h	01101000	150	104	68
	)	00101001	051	041	29	i	01101001	151	105	69
	*	00101010	052	042	2A	j	01101010	152	106	6A
	.	00101011	053	043	2B	k	01101011	153	107	6B
	,	00101100	054	044	2C	l	01101100	154	108	6C
	-	00101101	055	045	2D	m	01101101	155	109	6D
	.	00101110	056	046	2E	n	01101110	156	110	6E
	/	00101111	057	047	2F	o	01101111	157	111	6F
	0	00110000	060	048	30	p	01110000	160	112	70
	1	00110001	061	049	31	q	01110001	161	113	71
	2	00110010	062	050	32	r	01110010	162	114	72
	3	00110011	063	051	33	s	01110011	163	115	73
	4	00110100	064	052	34	t	01110100	164	116	74
	5	00110101	065	053	35	u	01110101	165	117	75
	6	00110110	066	054	36	v	01110110	166	118	76
	7	00110111	067	055	37	w	01110111	167	119	77
	8	00111000	070	056	38	x	01111000	170	120	78
	9	00111001	071	057	39	y	01111001	171	121	79
	:	00111010	072	058	3A	z	01111010	172	122	7A
	;	00111011	073	059	3B	{	01111011	173	123	7B
	<	00111100	074	060	3C	|	01111100	174	124	7C
	=	00111101	075	061	3D	}	01111101	175	125	7D
	>	00111110	076	062	3E	~	01111110	176	126	7E
	?	00111111	077	063	3F	DEL	01111111	177	127	7F

Index

End of Index