

Information Management Series  
**HP ALLBASE/4GL Developer  
Self-Paced Training Guide**

**For MPE/iX Systems**



HP Part No. 30601-64203  
Printed in USA 19920501

E0592

---

## NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD PROVIDES THIS MATERIAL "AS IS" AND MAKES NO WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE OR USE OF THIS MATERIAL WHETHER BASED ON WARRANTY, CONTRACT, OR OTHER LEGAL THEORY.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

SWT/SOFTWARE TECHNOLOGY CENTER  
8000 FOOTHILLS BOULEVARD  
ROSEVILLE, CA 95678, U.S.A.

1986, 1987, 1988, 1989, 1990, 1992 by HEWLETT-PACKARD COMPANY

FINAL TRIM SIZE : 7.0 in x 8.5 in

---

## Printing History

New editions of this manual incorporate all material updated since the previous edition. The listing below shows all edition dates for the manual.

First Edition	October 1988
Second Edition	September 1989
Third Edition	February 1990
Fourth Edition	May 1992

---

## Preface

### MPE/iX

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with other HP 3000 users you will encounter references to MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for the HP 3000s not based on the PA-RISC (precision architecture-reduced instruction set computing) architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as compatibility mode.

### About This Manual

This manual is a self-paced training guide for application developers who are learning how to use HP ALLBASE/4GL. It uses a number of tutorial style lessons to provide an introduction to the features and facilities available in HP ALLBASE/4GL. While following these lessons, you will build a small working application with HP ALLBASE/4GL.

The lessons in this manual are designed to allow you to complete this training course at your own pace, without additional classroom instruction or assistance from an instructor. We don't suggest any specific time or duration for this course. Instead, we suggest that you proceed at a pace that makes you feel comfortable. However, since later lessons do depend on the results of the earlier lessons, we recommend that you proceed through these lessons in the sequence presented in this manual.

This manual contains a total of 11 chapters. Chapters 1 through 6 provide an overview of the HP ALLBASE/4GL application, then demonstrate how to develop a simple application. Chapters 7 through 11 include lessons that let you build upon the basic application to create a more complex application that accesses multiple files.

This manual also contains a developer menu chart, an appendix, a glossary, and an index. The chart shows the structure of the HP ALLBASE/4GL developer menus and screens. The appendix outlines the procedure for defining a new application in the HP ALLBASE/4GL administrator.

If you are also the Administrator for your HP ALLBASE/4GL system, you will need to follow the procedures outlined in Appendix A before you can start working on the training lessons in this manual.

The glossary explains the meanings of some HP ALLBASE/4GL terminology. The index lists the application components and procedures you will find discussed in this manual.

# Contents

---

<b>1. Getting Started</b>	
Training Overview . . . . .	1-1
How Does HP ALLBASE/4GL Work? . . . . .	1-1
HP ALLBASE/4GL System . . . . .	1-3
The Operating System Interface . . . . .	1-3
Data Manager . . . . .	1-3
Administrator . . . . .	1-4
End User Applications . . . . .	1-4
Developer . . . . .	1-4
How the HP ALLBASE/4GL Developer Works . . . . .	1-4
The Application You Will Create . . . . .	1-5
How the Application Will Work . . . . .	1-6
Application Structure . . . . .	1-7
Before You Begin . . . . .	1-10
Choosing a Data Manager . . . . .	1-10
<b>2. Planning and Defining Your Application</b>	
Working with HP ALLBASE/4GL . . . . .	2-1
Phase 1—Application Planning . . . . .	2-2
Some Rules of Thumb for HP ALLBASE/4GL Names . . . . .	2-2
Naming Rules . . . . .	2-2
Case Sensitivity . . . . .	2-2
Phase 2—Data Definition . . . . .	2-3
Lesson 1 - Defining Field Specifications . . . . .	2-4
Objectives . . . . .	2-4
What is a Field Specification? . . . . .	2-4
Signing On to HP ALLBASE/4GL Developer . . . . .	2-5
Tips on Moving Around HP ALLBASE/4GL Screens . . . . .	2-6
Menus . . . . .	2-6
Data Screens . . . . .	2-7

Creating Field Specifications . . . . .	2-8
Using the Dictionary Field Specifications Screen . . . . .	2-8
If You Make a Mistake ... . . . .	2-13
Entry Notes . . . . .	2-15
HP TurboIMAGE/iX Field Specifications . . . . .	2-17
Summary . . . . .	2-17
Lesson 2 - Defining Record Layouts . . . . .	2-18
Objectives . . . . .	2-18
Record Layouts . . . . .	2-18
Task 1 - Creating a Record Layout Header . . . . .	2-19
Using the Record Layout Header Screen . . . . .	2-19
Task 2 - Defining Record Layout Details . . . . .	2-21
Using the Record Layout Details Screen . . . . .	2-21
Completing the Record Layout Details . . . . .	2-23
Entry Notes . . . . .	2-25
Task 3 - Generating a Record Layout . . . . .	2-28
Summary . . . . .	2-29
Lesson 3—Defining Data Files . . . . .	2-30
Objectives . . . . .	2-30
KSAM Data File Definition and Creation . . . . .	2-30
Task 1 - Defining a Data File . . . . .	2-31
Using the Data File/SQL Table Definition Screen . . . . .	2-31
Notes on Window Entries . . . . .	2-33
Task 2 - Creating the Data File . . . . .	2-34
Using the File/SQL Table Screen . . . . .	2-34
Summary . . . . .	2-36
What To Do Next . . . . .	2-36
HP ALLBASE/SQL Table Definition and Creation . . . . .	2-37
Task 1 - Defining an HP ALLBASE/SQL Table . . . . .	2-37
Using the Data File/SQL Table Definition Screen . . . . .	2-37
Task 2 - Creating an HP ALLBASE/SQL Table . . . . .	2-40
Using the Create File/SQL Table Screen . . . . .	2-40
Notes on SQL Tables . . . . .	2-42
Summary . . . . .	2-43
What To Do Next . . . . .	2-43
HP TurboIMAGE/iX Data Set Definition . . . . .	2-44
Task 1 - Defining an HP TurboIMAGE/iX Data Set . . . . .	2-44
Using the File Definition Screen . . . . .	2-45

**Contents-2**

Notes on Windows . . . . .	2-48
Summary . . . . .	2-48
Self Test Questions . . . . .	2-49
Answers . . . . .	2-51
<b>3. Developing Screens</b>	
How HP ALLBASE/4GL Screens Work . . . . .	3-2
Screen Types . . . . .	3-2
Menus . . . . .	3-2
Data Screens . . . . .	3-3
Windows . . . . .	3-3
Function Keys . . . . .	3-4
System Items . . . . .	3-4
Displaying Screens . . . . .	3-4
Lesson 4 - Creating Menus . . . . .	3-6
Objectives . . . . .	3-6
Developing a Menu . . . . .	3-6
Task 1 - Creating the the Menu Header . . . . .	3-7
Task 2 - Painting the Menu Details . . . . .	3-9
Using the Screen Painter to Paint a Menu . . . . .	3-9
Step 1 - Creating the Screen Title . . . . .	3-11
Some Useful Tips on Working with the Painter . . . . .	3-12
Using Layout Keys . . . . .	3-13
Using the Block Function . . . . .	3-15
Step 2 - Defining System Items for a Menu Screen . . . . .	3-16
Step 3 - Defining Actions for the Menu Screen . . . . .	3-20
Task 3 - Saving the Menu Screen . . . . .	3-22
Summary . . . . .	3-22
Lesson 5 - Creating Data Screens . . . . .	3-24
Objectives . . . . .	3-24
Defining Data Screens . . . . .	3-25
Task 1 - Defining the Screen Header . . . . .	3-25
Task 2 - Painting the Data Screen Image . . . . .	3-28
Step 1 - Entering Text Items . . . . .	3-28
Tips On Entering Text . . . . .	3-30
Step 2 - Entering the Screen Prompts . . . . .	3-30
Step 3 - Defining Data Fields . . . . .	3-31
Step 4 - Saving the Screen . . . . .	3-33

Task 3 - Specifying Screen Field Details . . . . .	3-34
Entry Notes . . . . .	3-42
Task 4 - Generating the Data Screen . . . . .	3-44
Summary . . . . .	3-45
<b>4. Defining Application Logic</b>	
Logic . . . . .	4-1
What Are Processes and Functions? . . . . .	4-1
What Are SQL Logic Blocks? . . . . .	4-2
Defining the Logic for the Training Application . . . . .	4-2
Lesson 6 - Developing a Process . . . . .	4-3
Objectives . . . . .	4-3
The Product Process . . . . .	4-4
Task 1 - Defining the Process Header . . . . .	4-4
Task 2 - Defining the Process Details . . . . .	4-6
About the Process . . . . .	4-7
Using Open Window Commands . . . . .	4-9
Creating an SQL Process . . . . .	4-15
Using Open Window Commands . . . . .	4-16
Generating the Process . . . . .	4-22
Notes on Entries . . . . .	4-22
Creating an HP TurboIMAGE/iX Process . . . . .	4-22
Using a Special Window Command . . . . .	4-29
Generating the Process . . . . .	4-32
Task 3 - Generating the Process . . . . .	4-32
Summary . . . . .	4-33
Lesson 7 - Developing a Function . . . . .	4-34
Objectives . . . . .	4-34
Functions . . . . .	4-34
Task 1 - Defining the Function Header . . . . .	4-35
Task 2 - Defining the Function Details . . . . .	4-36
Developing a KSAM Function . . . . .	4-37
Developing an SQL Function . . . . .	4-41
Developing an HP TurboIMAGE/iX Function . . . . .	4-45
Task 3 - Generating the Function . . . . .	4-49
Summary . . . . .	4-49
Lesson 8 - Creating an SQL Logic Block . . . . .	4-50
Objectives . . . . .	4-50



SQL Logic Blocks . . . . .	4-50
Defining an SQL Logic Block . . . . .	4-51
Defining the SQL Logic Block Details . . . . .	4-53
The SELECT Command . . . . .	4-54
Host Variables . . . . .	4-55
Generating an SQL Logic Block . . . . .	4-55
Generation Errors . . . . .	4-56
SQL Logic Block Limitations . . . . .	4-57
Another SQL Logic Block . . . . .	4-57
Summary . . . . .	4-58
Self Test Questions . . . . .	4-59
Answers . . . . .	4-60
<b>5. Testing the Application</b>	
Lesson 9 - Testing the Training Application . . . . .	5-2
Objectives . . . . .	5-2
Application Testing . . . . .	5-2
Any Problems? . . . . .	5-2
Using Trace Mode . . . . .	5-4
Handling Problems . . . . .	5-7
Problem: Items Not Found . . . . .	5-7
Problem: No Initial Action . . . . .	5-9
Lesson 10 - Using Developer Utilities . . . . .	5-10
Objectives . . . . .	5-10
The Utilities Menu . . . . .	5-10
Displaying the Catalog . . . . .	5-11
Copying Catalog Items . . . . .	5-13
Deleting Catalog Items . . . . .	5-14
Printing Catalog Items . . . . .	5-16
Generating Items . . . . .	5-18
Summary . . . . .	5-20
Understanding Screen Processing Logic . . . . .	5-21
Full Data Screen Processing . . . . .	5-21
Display Field Processing . . . . .	5-24
Input Field Processing . . . . .	5-25
Summary . . . . .	5-29
Data Movement . . . . .	5-30
Self Test Questions . . . . .	5-31

Answers . . . . .	5-32
<b>6. HP ALLBASE/4GL Reports</b>	
What is an HP ALLBASE/4GL Report? . . . . .	6-1
Format of a Report . . . . .	6-2
Line Groups . . . . .	6-2
Line Numbers . . . . .	6-3
Totalling Facilities . . . . .	6-3
The Product Report . . . . .	6-4
Lesson 11 - Defining the Report . . . . .	6-5
Objectives . . . . .	6-5
Task 1 - Creating the Report Header . . . . .	6-6
The Next Step for KSAM and TurboIMAGE/iX Applications	6-9
The Next Step for SQL Based Applications . . . . .	6-9
Task 2—Creating the Report Line Header . . . . .	6-10
Defining the Report Lines . . . . .	6-10
Summary . . . . .	6-16
Lesson 12 - The Report Painter . . . . .	6-17
Objectives . . . . .	6-17
Painting a Report . . . . .	6-17
Using the Window Keys . . . . .	6-19
Task 3 - Painting the Product Report . . . . .	6-20
Step 1 - Entering Text Items . . . . .	6-21
Step 2 - Defining Data Fields . . . . .	6-23
In Summary . . . . .	6-27
Creating Fields Using Dictionary Field Specifications . . . . .	6-28
Step 3 - Saving the Report . . . . .	6-29
Summary . . . . .	6-30
Lesson 13 - Generating and Testing a Report . . . . .	6-31
Objectives . . . . .	6-31
Generating a Report . . . . .	6-31
Testing a Report . . . . .	6-32
Testing by Menu Bypass . . . . .	6-32
Testing from a Menu . . . . .	6-33
Summary . . . . .	6-34
From Here On . . . . .	6-34
Self Test Questions . . . . .	6-35
Answers . . . . .	6-36

<b>7. Learning More Features</b>	
How Will the Modifications Look? . . . . .	7-2
The Application Screens . . . . .	7-3
Application Structure . . . . .	7-4
Lesson 13 - Creating Storage Items . . . . .	7-5
Objectives . . . . .	7-5
What Are Storage Items? . . . . .	7-5
Defining Application Titles . . . . .	7-6
Defining Variables . . . . .	7-8
Defining the Mode Variable . . . . .	7-9
Defining Numeric Constants . . . . .	7-10
KSAM Based and HP TurboIMAGE/iX Based Applications . . . . .	7-11
HP ALLBASE/SQL Based Applications . . . . .	7-12
Defining Alphanumeric Constants . . . . .	7-14
Creating Additional Alphanumeric Constants . . . . .	7-15
Summary . . . . .	7-16
Lesson 14 - Modifying a Logic Block . . . . .	7-18
Objectives . . . . .	7-18
Modifying the product_proc Process . . . . .	7-18
KSAM Based Applications . . . . .	7-18
Modifying a Process . . . . .	7-20
Step Number References . . . . .	7-21
Using the IF Window . . . . .	7-21
Continuing the Change . . . . .	7-24
Deleting a Step . . . . .	7-26
Modifying a Step . . . . .	7-26
Adding Commands . . . . .	7-28
Modifying HP ALLBASE/SQL Based Applications . . . . .	7-29
Modifying a Process . . . . .	7-30
Step Number References . . . . .	7-31
Using the IF Window . . . . .	7-32
Continuing the Change . . . . .	7-34
Adding Commands . . . . .	7-36
Deleting a Step . . . . .	7-36
Modifying a Step . . . . .	7-37
Modifying HP TurboIMAGE/iX Based Applications . . . . .	7-39
Modifying Logic Blocks . . . . .	7-40
Step Number References . . . . .	7-41

Using the IF Window . . . . .	7-42
Continuing the Change . . . . .	7-45
Adding Commands . . . . .	7-46
Deleting a Step . . . . .	7-46
Modifying a Step . . . . .	7-46
Summary . . . . .	7-48
Lesson 15 - Creating New Logic Blocks . . . . .	7-49
Objectives . . . . .	7-49
The Mode Functions . . . . .	7-49
Screen Field Logic . . . . .	7-50
Modifying KSAM Based Functions . . . . .	7-51
Modifying HP ALLBASE/SQL Based Functions . . . . .	7-53
Modifying HP TurboIMAGE/iX Based Functions . . . . .	7-55
File Manipulation Functions . . . . .	7-57
KSAM Based and HP TurboIMAGE/iX Based File Manipulation Functions . . . . .	7-57
The Add Function . . . . .	7-57
The Modify Function . . . . .	7-58
The Delete Function . . . . .	7-58
HP ALLBASE/SQL Based File Manipulation Functions . . . . .	7-59
The Add Function . . . . .	7-59
The Modify Function . . . . .	7-61
The Delete Function . . . . .	7-62
The Next Step . . . . .	7-64
Altering the Main Menu . . . . .	7-64
Painting the Screen Again . . . . .	7-64
Summary . . . . .	7-65
Lesson 16 - Creating Messages . . . . .	7-67
Objectives . . . . .	7-67
Messages . . . . .	7-67
Creating a Message . . . . .	7-69
Summary . . . . .	7-73
Lesson 17 - Validation Ranges . . . . .	7-74
Objectives . . . . .	7-74
Validation Range Definition . . . . .	7-74
Summary . . . . .	7-78
Self Test Questions . . . . .	7-78
Answers . . . . .	7-80

<b>8. Learning to Use the Module Builder</b>	
Lesson 18 - The Module Builder . . . . .	8-2
Objectives . . . . .	8-2
Necessary Preparations . . . . .	8-3
Defining the Field Specifications . . . . .	8-3
Creating the Record Layout . . . . .	8-6
Creating the File Specification . . . . .	8-6
Using the Module Builder Screen . . . . .	8-8
Using the Module Details Screen . . . . .	8-11
Linking a Field to a Secondary File . . . . .	8-13
Inserting a Display Field . . . . .	8-15
Final Steps . . . . .	8-16
Module Generation . . . . .	8-16
The Module Building Process . . . . .	8-16
Joining the Module to the Main Menu . . . . .	8-17
Further Options . . . . .	8-17
Running the Module . . . . .	8-18
Using the Module . . . . .	8-18
Retrieving an Existing Record . . . . .	8-19
Summary . . . . .	8-20
Necessary Preparations . . . . .	8-20
Module Building . . . . .	8-20
Joining the Module to the Application . . . . .	8-21
Running the Module . . . . .	8-21
Conclusion . . . . .	8-21
Self Test Questions . . . . .	8-22
Answers . . . . .	8-23
<b>9. Learning Additional Screen Techniques</b>	
About the Enhanced Application . . . . .	9-2
Product/Option Application . . . . .	9-2
Function Keys . . . . .	9-3
Scrolling Options . . . . .	9-3
Reporting . . . . .	9-3
Processing Rules . . . . .	9-4
Application Structure . . . . .	9-4
Product/Option Application . . . . .	9-4
Lesson 19 - Adding Dictionary Items . . . . .	9-6

Objectives . . . . .	9-6
Dictionary Requirements . . . . .	9-6
Task 1 - Creating Dictionary Field Specifications . . . . .	9-7
Task 2 - Creating the Record Layout and File . . . . .	9-8
Task 3 - Creating Variables . . . . .	9-11
Task 4 - Creating Alphanumeric Constants . . . . .	9-12
Task 5 - Creating Numeric Constants . . . . .	9-12
Task 6 - Creating Validation Tables . . . . .	9-13
Summary . . . . .	9-16
Lesson 20 - More Screen Techniques . . . . .	9-17
Objectives . . . . .	9-17
Screens for the Enhanced Training Application . . . . .	9-18
Task 1 - Altering the Product Data Screen . . . . .	9-18
Task 2 - Creating the Option Window . . . . .	9-22
Task 3 - Adding a new Option Window . . . . .	9-25
Task 4 - Creating the The prod_wnd Window . . . . .	9-26
Summary . . . . .	9-27
Lesson 21 - Adding Function Keys . . . . .	9-28
Objectives . . . . .	9-28
Function Keys . . . . .	9-28
Defining Function Keys . . . . .	9-29
Defining a Second Function Key Set . . . . .	9-32
Defining Function Key Logic . . . . .	9-33
Summary . . . . .	9-35
Lesson 22 - Scrolling Data on the Screen . . . . .	9-37
Objectives . . . . .	9-37
Understanding the Scrolling System . . . . .	9-37
Task 1 - Creating Secondary File Record Layouts . . . . .	9-37
Creating HP ALLBASE/SQL Select Lists . . . . .	9-39
Defining a Select List . . . . .	9-39
Generating a Select List . . . . .	9-41
Task 2 - Creating The Scrolling Functions . . . . .	9-42
Creating the Functions Called from the scroll_options	
function . . . . .	9-45
Creating the last Function . . . . .	9-48
Creating HP ALLBASE/SQL Functions . . . . .	9-50
Creating HP TurboIMAGE/iX Functions . . . . .	9-55
Summary . . . . .	9-59

**Contents-10**

Lesson 23 - Creating User Help Screens . . . . .	9-61
Objectives . . . . .	9-61
Defining Help Screens . . . . .	9-61
Creating Field Level Help . . . . .	9-62
Creating Message Level Help . . . . .	9-63
Creating Screen Level Help . . . . .	9-63
Summary . . . . .	9-64
Self Test Questions . . . . .	9-64
Answers . . . . .	9-66

**10. Expanding Logic Blocks**

Lesson 24 - Extending the Application Logic . . . . .	10-1
Objectives . . . . .	10-1
Modifying the product_proc Process . . . . .	10-2
Understanding the Modifications to the Logic . . . . .	10-3
Modifying the product_key_read Function . . . . .	10-6
Creating the option_key_read Function . . . . .	10-14
Creating Messages for the Logic Blocks . . . . .	10-22
Summary . . . . .	10-23
Lesson 25 - Using Decision Tables . . . . .	10-24
Objectives . . . . .	10-25
Defining the Decision Table Header . . . . .	10-25
Decision Table Questions . . . . .	10-27
Defining Decision Table Actions . . . . .	10-29
Defining Decision Table Relationships . . . . .	10-31
Testing the Modified Application . . . . .	10-37
Summary . . . . .	10-38
Lesson 26 - Creating the File Manipulation Functions . . . . .	10-39
Objectives . . . . .	10-39
Defining the Add Functions . . . . .	10-39
Modifying the option_number Function . . . . .	10-43
Creating the Appropriate Messages . . . . .	10-45
Creating the modify_option Function . . . . .	10-45
Creating the Delete Functions . . . . .	10-48
Creating the Delete Functions for KSAM Based Applications . . . . .	10-48
Modifying the delete_product function . . . . .	10-48
Modifying the del_all_options function . . . . .	10-51
Creating the del_corrupt_prod function . . . . .	10-54

Deleting Options . . . . .	10-56
Creating Functions for HP ALLBASE/SQL Based Applications	10-58
Deleting a Product . . . . .	10-58
Modifying SQL Logic Blocks . . . . .	10-60
Deleting Options . . . . .	10-62
Creating the Functions for HP TurboIMAGE/iX Based	
Applications . . . . .	10-64
Deleting Options . . . . .	10-72
Summary . . . . .	10-73
Self Test Questions . . . . .	10-74
Answers . . . . .	10-75

**11. Developing Greater Reporting Capabilities**

Lesson 27-Learning More Reporting Techniques . . . . .	11-1
Objectives . . . . .	11-1
Creating The Product/Option Report . . . . .	11-2
Some Preparations . . . . .	11-2
Creating the HP ALLBASE/SQL Based Applications Select	
List . . . . .	11-2
Defining the Report Header . . . . .	11-3
Creating the Screen . . . . .	11-5
Creating the Function . . . . .	11-6
Defining Report Sorting Levels . . . . .	11-10
Defining Report Selection Criteria . . . . .	11-13
Line Headers . . . . .	11-16
KSAM Based and HP TurboIMAGE/iX Based Applications .	11-17
HP ALLBASE/SQL Based Applications . . . . .	11-18
Report File Linkages . . . . .	11-19
Defining the Functions . . . . .	11-22
Painting the Report . . . . .	11-24
KSAM Based and HP TurboIMAGE/iX Based Applications .	11-24
Description of the Subtotal Lines . . . . .	11-27
Creating the Final Total Group . . . . .	11-28
HP ALLBASE/SQL Based Applications . . . . .	11-29
Understanding Subtotal Lines . . . . .	11-31
Summary . . . . .	11-32
Lesson 28-Defining Calculated Items . . . . .	11-35
Objectives . . . . .	11-35

**Contents-12**



Calculated Items . . . . .	11-35
Generating the Report . . . . .	11-39
Altering the Main Menu . . . . .	11-39
Running the Report . . . . .	11-40
Summary . . . . .	11-40
Where You Are Now . . . . .	11-41
Self Test Questions . . . . .	11-42
Answers . . . . .	11-43
<b>A. Starting a New Application</b>	
Before You Start . . . . .	A-1
The Sign-On Screen . . . . .	A-2
Administrator Sign-On . . . . .	A-2
Administrator Main Menu . . . . .	A-3
Developer Validation . . . . .	A-3
Training Application Definition . . . . .	A-5
Application Password Definition . . . . .	A-6
Development Security Code . . . . .	A-7
Other Fields . . . . .	A-8
Users and Groups List . . . . .	A-9
SQL Application Definition . . . . .	A-10
SQL Database Definition . . . . .	A-10
Database Access . . . . .	A-11
HP TurboIMAGE/iX Database Creation . . . . .	A-12
Defining the Database . . . . .	A-14
The Parameters for Database Access Screen . . . . .	A-16

**G. Developer Reference Chart**

**Glossary**

**Index**

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Getting Started

---

### Training Overview

Welcome to the *HP ALLBASE/4GL Self-Paced Training Guide*. This guide will help you become familiar with using HP ALLBASE/4GL to develop applications.

The lessons in this guide will lead you through the tasks required to create a basic application using the tools within HP ALLBASE/4GL. Once you have completed the lessons you will find that you can go on to create more complex applications with ease.

This guide assumes that you are already familiar with a conventional programming language, although it does not assume you have any experience with a programming productivity tool like HP ALLBASE/4GL.

---

### How Does HP ALLBASE/4GL Work?

HP ALLBASE/4GL is an advanced fourth-generation programming language. It enables you to design and implement application software by defining the required results, rather than defining the procedures necessary to achieve those results.

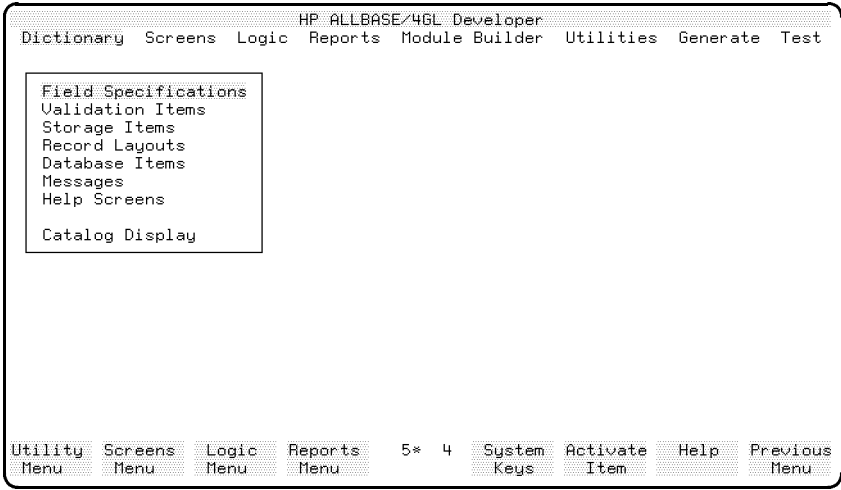
You can develop new applications, modify existing applications, and produce multiple versions of existing applications without having to develop the lines of code required for those applications.

HP ALLBASE/4GL consists of menus and screens which act as templates on which you input your application requirements such as field specifications and record layouts. HP ALLBASE/4GL then creates the application functions and screens for you.

Function key labels are included at the bottom of every screen. Each label corresponds to a function key located along the top row of your keyboard.

Function keys make it easier for you to move around within HP ALLBASE/4GL. They include function keys that access the system, keys that let you move from screen to screen, and keys that help you work with the various screens used to develop an application. You also can create function keys for use by the application's end user.

Shown below is a sample HP ALLBASE/4GL screen. This is a menu screen that lists some of the various categories of screens available to you as developer. At the bottom of the screen are the function keys that can be used with this screen. Note that the **System Keys** function key leads you to additional function keys. For a thorough discussion of function keys used in HP ALLBASE/4GL, refer to your *HP Developer Reference Manual*.



**Dictionary Menu Used in Developing Applications**

**1-2 Getting Started**

---

## HP ALLBASE/4GL System

HP ALLBASE/4GL is available in two versions: the full developer system and the run-time environment. The developer system contains all the facilities necessary to develop and run applications. The run-time environment provides the necessary facilities for application end users to run developed applications.

HP ALLBASE/4GL consists of five major functional components:

- Operating System Interface
- Data Manager
- Administrator
- End User Applications
- Developer

The run-time environment includes all functional components of HP ALLBASE/4GL except the developer application.

### The Operating System Interface

HP ALLBASE/4GL operates on Hewlett-Packard HP 3000 series 900 computer systems using the MPE/iX operating system.

The HP ALLBASE/4GL operating system interface handles all the communication with the host operating system. For the most part, you will never need to use MPE/iX commands while developing applications. If the need arises, however, you can use the MPE/iX system to run external programs in a language other than HP ALLBASE/4GL. You can call such external programs from within HP ALLBASE/4GL.

### Data Manager

The data manager handles all access to application databases. The data manager allows multiple users and multiple applications to access the same database simultaneously.

A built-in KSAM (Keyed Sequential Access Method) data file manager provides the facilities to create and delete KSAM data files, and you can also use serial files in an HP ALLBASE/4GL application.

The data manager provides access to HP ALLBASE/SQL databases and HP TurboIMAGE/iX within HP ALLBASE/4GL.

## **Administrator**

The administrator is an HP ALLBASE/4GL application. Your system administrator uses the administrator application to define system user names, applications and versions, and various system wide defaults.

## **End User Applications**

HP ALLBASE/4GL supports two types of end user applications: base applications and versions.

A base application contains all the logic required to run that application. A version uses the underlying base application logic but includes an additional set of definitions that modify it for a particular end user or group of end users. A version cannot run unless its base application exists on the same system.

## **Developer**

The HP ALLBASE/4GL developer is also an HP ALLBASE/4GL application. This is the application that you use to create end user applications. As you work through the lessons in this guide, you will be using the developer facilities to develop a working application.

---

## **How the HP ALLBASE/4GL Developer Works**

The HP ALLBASE/4GL developer provides you with application development facilities that can be grouped under the following headings:

- Dictionary.
- Screen development.
- Logic.
- Reports.

### **1-4 Getting Started**

You will work with each of these facilities in developing the sample application.

The developer application uses menus and screens to guide you through each phase of the development process. You specify most application items by filling in formatted screens to define the details of the item.

As you develop an application, your definitions are stored in a set of internal files. The application definitions exist in two forms: application “source” and the “generated” application. Conceptually, the source application is similar to source code and the generated application is similar to object code in a conventional programming system.

The generated form is an executable form of the application. HP ALLBASE/4GL uses the instructions in the generated application to present the application items to the user. The application source code does not need to be present at run-time, although its presence makes no difference.

The developer environment contains all the logic necessary to convert your application source code to the generated form. You don't need any external compilers or utilities.

Both the developer system and the run-time environment contain all the necessary facilities to run generated applications.

---

## The Application You Will Create

You can create a wide variety of types of applications using HP ALLBASE/4GL. Following is a description of the *training* application, the one you will create using this training guide.

The *training* application is a simplified product listing. It consists of one menu and one screen. Four data items have been identified for the application: product number, description, supplier number, and lead time. For each of those items you will do the following:

- Create dictionary definitions, record layouts, and data files
- Create the menu and screen
- Define the logic required

- Design a report for the application
- Test the application to make sure it runs

## How the Application Will Work

The initial action in an application can be either a menu or a process. In this application, the initial action is the menu. When the user signs on to the *training* application, HP ALLBASE/4GL will automatically display the menu for the application.

When the user selects the *Product Details* option on the main menu, the application will execute a process called *product\_proc*. This process displays the *product\_scrn* screen.

The screenshot shows a terminal window titled "Training Application" with a timestamp of "14:38:10". The main content area is titled "Product Details" and "product\_scrn" with a date of "3/19/92". It contains four data entry fields: "Product Number", "Description", "Supplier Number", and "Lead Time". At the bottom, there is a menu bar with the text "14\* 35" and four menu items: "System Keys", "Commit Data", "Help", and "Previous Menu".

**The product\_scrn Screen**

The four data fields defined for the application are shown above on the application's *product\_scrn* screen.

The *product\_scrn* screen will have a function associated with the first field. After the user has completed data entry in the product number field, this function reads the product file for a record matching the product number just

## 1-6 Getting Started



entered. If a matching record does not exist, the user can enter the details for a new record.

If a matching record exists, the function displays the details of the record on the screen. The user can now modify the details of the record.

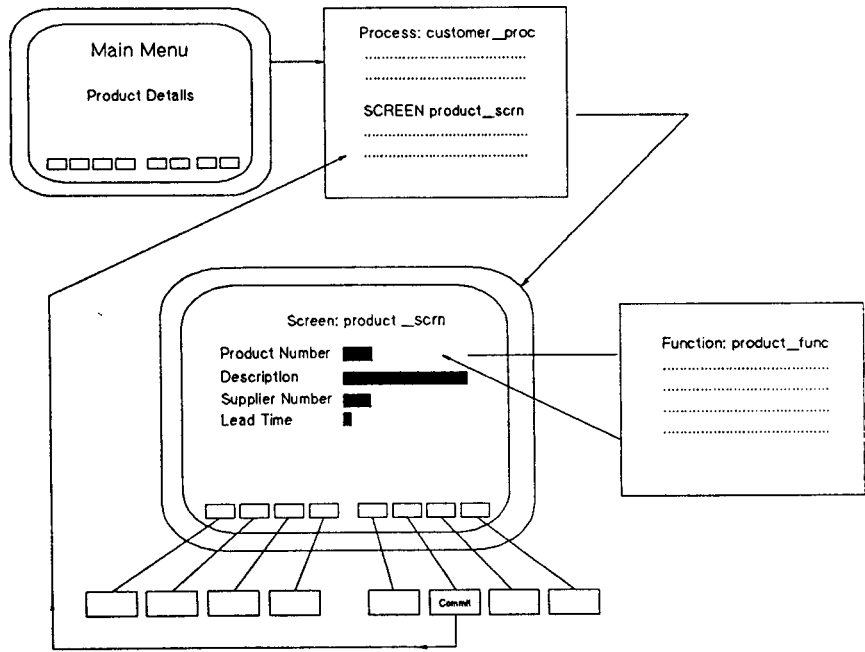
When the user presses the **Commit Data** function key, control returns to the process. The process updates the product file to reflect the new or changed data entered by the user, and then displays the screen again.

To exit from the process, the user must press the **Previous Menu** function key.

## **Application Structure**

The underlying structure of the application can be represented as shown in the following diagram.

You can consider an application as being a series of linked elements. In this case, the sequence of elements is a process, a screen, and then a function. All the elements in a sequence such as this are identified by a name.



### Application Structure

One standard format for constructing applications is to use a process to call a screen, and, on exit from the screen, return to the process to update one or more files to reflect data entered while the screen is active. A function associated with a screen field can be used to perform a file lookup while the screen is active.

This structure emphasizes the similarity between an HP ALLBASE/4GL process and a program in a conventional language, and the similarity between an HP ALLBASE/4GL function and a conventional language subroutine.

When a process starts, it initializes the environment to a known state. This means that all application data files are closed; all file record buffers, work areas, and screen buffers are cleared; any current process, function, screen, decision table, or report is terminated; and any incomplete HP ALLBASE/SQL transaction is reversed. Using a process to control the screen and subsequent

### 1-8 Getting Started

file update means that the screen display and file update occur within a known and controlled environment.

An HP ALLBASE/4GL application only allows one process to operate at a time. In this case, the process remains active and controls the application until the user terminates it by pressing the **Previous Menu** function key.

The function associated with the first field on the screen behaves somewhat like a subroutine. It performs a file lookup, and the application then continues from the point where the function was called after the file lookup is complete. The function does not change the overall environment for the application.

The flow of control on exit from different types of application items operates according to the following rules.

<b>Item</b>	<b>Control returns to ...</b>
Process	The last menu displayed by the application. If no menu has been displayed, the application terminates.
Function	The item that called the function.
Data screen	The next command in the logic block that called the screen, or the previous menu if the screen was called from a menu.

This application structure will become apparent as you perform the tasks needed to develop the *training* application. This application was designed to help you become familiar with HP ALLBASE/4GL and to demonstrate the logic and data processing capabilities associated with its screens. In chapter 2 we will begin the process of developing the *training* application.

---

## Before You Begin

Before you can start the training process described in this guide you must:

- be able to sign on to HP ALLBASE/4GL as a developer.
- have the *training* application defined in the *administ* application.
- have chosen the data manager you will use for the *training* application.

The process you use to start HP ALLBASE/4GL may vary from one installation to another. You may need to talk to your system administrator about the procedure to log in to the MPE/iX system, and to start HP ALLBASE/4GL. Your HP ALLBASE/4GL administrator will allocate an HP ALLBASE/4GL user name for you so you can sign on as a developer.

If you are the system administrator, refer to the *HP ALLBASE/4GL Developer Administration Manual* to find out how to install HP ALLBASE/4GL. That manual also describes the procedures for defining HP ALLBASE/4GL user names and application names.

An application cannot be developed in HP ALLBASE/4GL until it has been defined in the *administ* application. To define the *training* application, you or your system administrator should follow the instructions in Appendix A of this guide. This will enable you to develop and run the *training* application.

## Choosing a Data Manager

HP ALLBASE/4GL allows you to create applications that interface with a number of data managers: KSAM data files, HP ALLBASE/SQL databases, HP TurboIMAGE/iX databases, and serial data files.

This training guide includes instructions for using the first three listed data managers to build your application. Most of the time, there is no difference in the instructions for the three applications. In some cases, however, especially when you are manipulating data in a file, table, or data set, there may be slight differences. When this occurs, separate instructions for that database are provided.

## 1-10 Getting Started

---

**Note**

To build the *training* application using the HP ALLBASE/SQL or HP TurboIMAGE/iX databases, you must have that database installed on your system.

---

When a difference between data managers occurs, this training guide will direct you to read the appropriate instructions. You can skip the sections that are irrelevant to you.

If you are not sure about which data manager to use for your application, ask your HP ALLBASE/4GL administrator for advice or refer to your *HP ALLBASE/4GL Developer Reference Manual*.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Planning and Defining Your Application

---

### Working with HP ALLBASE/4GL

HP ALLBASE/4GL does not place many restrictions on the way you can develop applications, apart from the need to define dictionary items before they can be used in other parts of the application.

Typically, developing an application with HP ALLBASE/4GL involves five phases:

- Phase 1—Application planning.
- Phase 2—Data definition.
- Phase 3—Screen development.
- Phase 4—Logic definition.
- Phase 5—Report definition.

These phases represent one of many possible approaches to application development using HP ALLBASE/4GL. We will examine each one in turn as you develop the sample application.

Each phase and task you perform includes an explanation to aid you in understanding its purpose. For more detailed information about any of the phases or the tasks they require, refer to your *HP ALLBASE/4GL Developer Reference Manual*.

---

## Phase 1—Application Planning

The first task in application planning is to define the requirements of your application. As part of this task, you need to identify the field specifications, record layouts, and data files the application needs.

For this training guide, that step has been done for you. The type of data required for each field has been defined, and the number of menu and data screens has been determined.

### Some Rules of Thumb for HP ALLBASE/4GL Names

Although the names for the *training* application have already been selected, you should be aware of the following general rules regarding names used in HP ALLBASE/4GL applications.

#### Naming Rules

- All names must start with an alphabetic character. Names can contain any combination of alphabetic characters, 0 to 9, and \_ (underscore). Components of KSAM based and HP TurboIMAGE/iX based applications may also include extended (eight-bit) characters.
- The names that you use must be unique for each type of item that you are creating. However, you can use one name for a number of different types of items. For example, you can have a report, a screen, a field and a process all called *order*. However, you cannot have two reports with the name *order*.
- You can use up to eight characters for:
  - modules built by the module builder,
  - file names or SQL tables names, and
  - SQL select list names.
- You can use up to 16 characters to name all other application items.

#### Case Sensitivity

HP ALLBASE/4GL names are case sensitive, so you must always type the name of an item the same way each time that you use it. For example, the variable *V-item\_cost* is not the same as the variable *V-ITEM\_COST*.

## 2-2 Planning and Defining Your Application



HP ALLBASE/SQL and HP TurboIMAGE/iX, on the other hand, are not case sensitive; they automatically shift some HP ALLBASE/4GL names to uppercase.

This means that you must make sure that the record layout does not contain two field specifications that have the same name when they are shifted to uppercase. The field specification names *Account\_No* and *account\_no* are distinct in HP ALLBASE/4GL, but are duplicates in HP ALLBASE/SQL or HP TurboIMAGE/iX databases.

HP ALLBASE/4GL displays a warning if you attempt to define a table or data set name, and the same name already exists in uppercase format.

---

## Phase 2—Data Definition

This chapter contains three lessons that explain the requirements for defining data and demonstrates the use of HP ALLBASE/4GL screens to input those requirements.

- In lesson 1 you will learn how to define the fields for the *training* application.
- In lesson 2, you will define the record layouts.
- In lesson 3 you will define the files for the completed field definitions and allocate physical file space for each field.

The HP ALLBASE/4GL screens you will learn how to use include:

- Dictionary Field Specifications Screen
- Record Layout Header Screen
- Record Layout Details Screen
- File/SQL Table Definition Screen

---

## Lesson 1 - Defining Field Specifications

---

### Objectives

When you have completed this lesson, you will have learned:

- The characteristics of field specifications
- How to use the field specifications screen to specify fields

### What is a Field Specification?

A dictionary field specification is a template that defines the characteristics of a “field.” A field is a data item that can exist on a file, a screen, or a report. A field specification defines the attributes such as name, length, and type, of a field used throughout the application. A field on any file, screen, or report can refer to a field specification in the dictionary.

When you make a change to a field specification, the changes propagate through the application to all screens, reports and file definitions.

HP ALLBASE/4GL allows you to define the following field specification parameters:

- The field specification name.
- Whether the field specification is secured or not.
- How long the field is, and whether it is repeated or not.
- If the field has a minimum entry length.
- What the edit code of the field is, and how it is justified.
- What pad character is used for the field.
- If there are any decimal places associated with the field.
- If a numeric field is displayed as blank when it only contains zeros.
- How data is physically stored in a field.
- How data entered for the field is validated.

### 2-4 Planning and Defining Your Application

HP ALLBASE/4GL allows you to define help screens for the fields you are specifying and enter descriptive documentation that will be helpful when you review your application.

Now it is time to sign on to the HP ALLBASE/4GL application.

---

## Signing On to HP ALLBASE/4GL Developer

The process you use to start HP ALLBASE/4GL may vary from one installation to another. You may need to talk to your system administrator about the procedure to log in to the MPE/iX system and to start HP ALLBASE/4GL. Your system administrator will allocate an HP ALLBASE/4GL user name for you so you can sign on as a developer.

Once you have your user name, start the HP ALLBASE/4GL developer program. The default command is:

Run `hp4gl`

The sign on screen will appear.



**Developer Sign On Screen**

**To activate the HP ALLBASE/4GL developer application:**

1. At the sign on screen, enter the name *developr* and press **Return**. An additional field will appear.
2. Enter *training* in the *Application or Version to be developed* field.
3. Press the **Commit Data** function key.

This action causes the developer main menu to appear. You are ready to begin physically developing the *training* application.



**Developer Main Menu**

**Tips on Moving Around HP ALLBASE/4GL Screens**

**Menus**

When you first access a menu, the first item on the menu is highlighted. You press **Return** or the **Activate Item** function key to execute the highlighted item.

You can use the **Tab** key to select other items on the menu. Each time you press the **Tab** key, the highlight moves to the next menu item.

**2-6 Planning and Defining Your Application**

You can also use the cursor keys `cursor left`, `cursor right`, `cursor up` and `cursor down` to select menu items. To use these keys, move the cursor to the item you want to select and press `Return`. HP ALLBASE/4GL responds by highlighting the item at the current cursor position. Press `Return` to execute the item. You can only highlight items that are on the active menu. If you attempt to highlight an item outside the active menu, HP ALLBASE/4GL will highlight the item on the active menu that is closest to the current cursor position.

On any menu, pressing `cursor home` followed by `Return` takes you to the first item on the menu. Pressing `Shift` and `cursor home` followed by `Return` takes you to the last item on the menu.

## Data Screens

Data screens in the developer are the screens that allow you to enter data to define the details of application components. In data screens, you can use a number of the terminal keys while entering data in fields or moving about the screen.

You can use any of the `Clear line`, `Insert char`, `Delete char`, `Back space`, `cursor right` or `cursor left` keys to edit the data entered or to move about within a data input field on a data screen.

A number of keys allow you to move from one field to another on a data screen. The `Return` key terminates the current field and moves the cursor to the next field in the screen tabbing sequence. The `Tab` key behaves the same way. Pressing the `Shift` and `Tab` keys together moves the cursor to the start of the current field. If the cursor is at the start of a field, pressing the `Shift` and `Tab` keys together, and then pressing `Return` moves the cursor to the previous field in the screen tabbing sequence.

You can use the `cursor up`, `cursor down`, `cursor left` and `cursor right` keys to move the cursor to a field on the screen. Press `Return` when you have moved the cursor to the field. HP ALLBASE/4GL then highlights the field and you can enter data into the highlighted field.

To move the cursor to the first field on the screen, press `cursor home` and then press `Return`. To move the cursor to the last field on the screen, press `Shift` and `cursor home` together and then press `Return`.

Moving the cursor from one field to another field and then pressing `Return` initiates a *field level commit* action for the first field.

---

## Creating Field Specifications

The four data fields you will need for the *training* application have the following contents:

<b>Data Field</b>	<b>Contents</b>
Product Number	6-character identifier
Description	30-character free form product description
Supplier Number	6-digit identifier
Lead Time	2-digit value

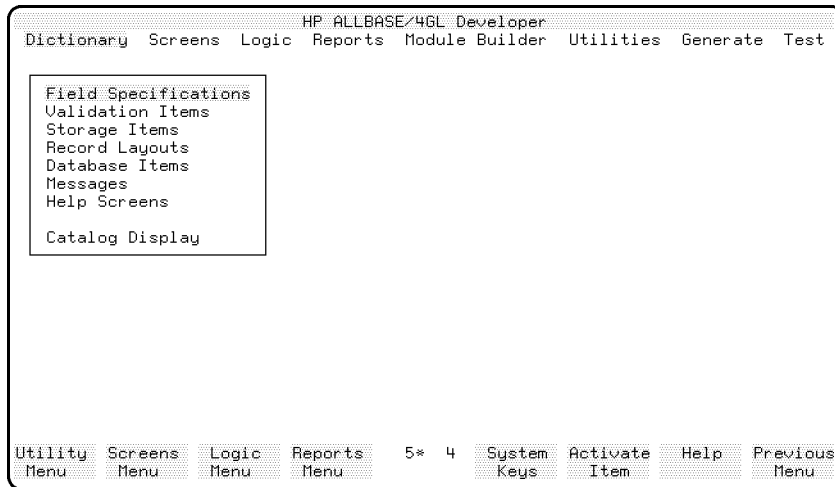
### Using the Dictionary Field Specifications Screen

The dictionary field specifications screen lets you define a field specification that is then used as a template in the records, screens, and reports for this application.

It is important to remember that defining a field specification only defines the essential characteristics of a field; it does not allocate any storage space for the field. That part of the development process takes place later.

#### To access the field specifications screen:

1. At the main menu screen, press the **Activate Item** function key.  
HP ALLBASE/4GL displays the dictionary menu. The first option, *Field Specifications* is highlighted.



### Dictionary Menu

2. Press the **Activate Item** function key again to activate the item *Field Specifications*.

You can only activate a highlighted item. If the *Field Specification* option is not highlighted, use the tab key to move up or down the list until the item is highlighted.

The next screen shows the field specifications screen as it will appear when you have entered all the screen details for the first field, the product number field.

Developer		Field Specifications		field_specs	
Field Spec. Name	product_no	Secured	N (Y/N)		
Field Length	6	Repeated	1	Times	
Minimum Entry Length	6				
Edit Code	U (X/A/U/K/N/S/Q/D/T)	Decimal Places			
Storage Type	C (C/I/L/F/G/P)	Blank When Zero		(Y/N)	
Justification	L (L/R/C/N)	Pad Character			
Validation: Range		Table			
Help Name					
Description	product_no				
AUTHOR:	developr				
	This field forms a unique six-character product identifier.				
Last Modification:	Date	4/ 8/92	Time	12:02:17	
Records Menu	Ranges	Tables	Data Mgr Attribs.	20* 34	System Keys
					Commit Data
					Help Previous Menu

### Field Specifications Screen

The first field specification you are going to define is for the product number field. To do this,

- Complete the listed fields on the field specification screen.

The entries for each field are listed for you. An explanation of each field is included.

- Complete the entry of data into a screen field by pressing **Return** or **Tab**.

HP ALLBASE/4GL then accepts the current contents of the field, and moves the cursor to the next field.

## 2-10 Planning and Defining Your Application



**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Field Spec. Name</b>	product_no	The name used to refer to the field specification within the application.
<b>Secured</b>	Press <b>(Return)</b> to accept the default of <b>N</b> .	This indicates whether the item is secured against modification by unauthorized developer.
<b>Field Length</b>	6	Maximum number of characters allowed for this field. For character and packed decimal field types, this also specifies the space the field occupies when used in a file record, screen, or report.
<b>Repeated Times</b>	Accept the default of 1.	When a field is one of several fields, as in a single dimensional array, you can specify the number of times the field is repeated. Not allowed in SQL applications.
<b>Minimum Entry Length</b>	6	Specifies the minimum number of characters to be entered. Indicates a required field. By entering <b>6</b> , you are ensuring that only a six character entry is valid.
<b>Edit Code</b>	Program skips this field.	Indicates type of data field can contain. <b>U</b> indicates forced uppercase. All entries will be shifted automatically to uppercase.
<b>Storage Type</b>	Accept the default.	Indicates the way data in the field is physically stored in a file. <b>C</b> indicates <b>character</b> . For a list of storage types, refer to the "Entry Notes" section that follows this one.
<b>Justification</b>	Accept default.	Indicates the direction of justification applied when a value is entered into or displayed in this field. <b>L</b> means left justified, and is the default entry for a character edit code.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Decimal Places</b>	Cursor skips this field.	Indicates maximum number of decimal places. For an integer, the number of decimal places is set to zero.
<b>Blank When Zero</b>	Cursor skips this field.	When the edit code field entry is <b>U</b> , the storage type is automatically <b>C</b> for “character,” which means the field will be blank.
<b>Pad Character</b>	Accept default (a space).	Specifies the single character used to fill out a field when field contains less than maximum number of characters specified.
<b>Range</b>	Press <b>Return</b> .	Method of validating the value a user enters; done automatically.
<b>Table</b>	Press <b>Return</b> .	Method of validating the value a user enters; done automatically.
<b>Help Name</b>	Press <b>Return</b> .	Not required. An entry here is written into the field definition screen in the screens menu. Name of context-sensitive help screen for this field.
<b>Description (Field Name)</b>	Accept default.	First line defaults to name of item being defined.
<b>Description (Author)</b>	Enter your name.	Defaults to <b>developr</b> .
<b>Description</b>	Enter a description of the field for documentation purposes.	Contents of description fields are used when a report is printed. They help keep track of items defined and their purpose in the application. A suggested entry is displayed on the completed field specifications screen shown earlier.
<b>Last Modification</b>	No entry	Automatically updated to show the time and date of the last modification to any of the fields on the screen.

For information on edit codes and storage type codes, refer to the “Entry Notes” at the end of this instructional segment.

## 2-12 Planning and Defining Your Application

### **If You Make a Mistake ...**

If you make a mistake, use the cursor keys, the **Tab** key, or the **Shift** + **Tab** keys to move to the field containing the error and make your corrections. To move to a previous field, press the **Shift** key with the **Tab** key, then press the **Return** key.

### **To complete and commit the screen:**

1. When you have completed the entries on this screen and all fields contain the correct values, press the **Commit Data** function key to actually create the field specification.

HP ALLBASE/4GL creates the field specification that you have just defined, clears the screen, and returns the cursor to the *Field Spec. Name* field so you can enter a new field specification name.

You can now define the rest of the field specifications for the product details.

### **To complete the remaining field specifications:**

1. For the three remaining field specifications, enter the values shown on the next page for the fields that require an entry.
2. For all other fields, simply press the **Return** or **Tab** key to accept the default.
3. Enter text in the description fields
4. Press the **Commit Data** function key after entering each field specification.

To complete the Description Field entries:

Field	Entry	Explanation
Field Specification	description	Name used to refer to the field specification within the application.
Field Length	30	This is a 30-character field.
Edit Code	X	Indicates an alphanumeric field.
Description	Enter a description for documentation purposes.	

To complete the Supplier Number Field entries:

Field	Entry	Explanation
Field Spec. Name	supplier_no	
Field Length	6	
Edit Code	N	Means that the entry must be an unsigned number (0 to 9, a period ., or a comma ,).
Storage Type	Enter I	Means that data will be stored as a two-byte integer.
Blank when Zero	Y	Screen displays blank for numeric value of zero. An N entry would display the zero, which can be distracting on the screen.
Description	Enter a description for documentation purposes.	

## 2-14 Planning and Defining Your Application

To complete the Lead Time Field entries:

Field	Entry	Explanation
Field Spec. Name	lead_time	
Field Length	2	This is a 2-character field.
Edit Code	N	Indicates a numeric field.
Storage Type	I	
Blank when Zero	Y	Screen displays blank for numeric value of zero. An N entry would display the zero, which can be distracting on the screen.
Description	Enter a description for documentation purposes.	

This completes the entries for the field specifications screen.

### Entry Notes

#### Edit Codes

The available edit codes for field specifications are shown in the following table:

Code	Meaning
X	Any printable character.
A	Alphabetic characters only (A to z, 0 to 9, extended characters, and underscore).
U	Forced uppercase. Same as X edit code, but shifts all lowercase alphabetic characters to uppercase.
K	Alphanumeric characters only (A to z, 0 to 9, extended characters, and underscore).

**Code    Meaning**

- N    Unsigned number (0 to 9, an period “.”, or a comma “,”).
- S    Signed number. Same as N edit code, but may include + or -.
- Q    Question. Y, y, N, or n only. Any characters beyond the first character in the field are ignored.
- D    Date field. Must be eight characters long. The date may be either MM/DD/YY or DD/MM/YY (depending on the system-wide date format) where MM, DD, and YY represent the month, day and year respectively.
- T    Date field, defaulting to the current date.

For on-line help about edit codes, press the **Help** function key. All input fields in the developer have a help screen that you can access while the cursor is on that field.

**Storage Type Codes**

The following table lists the available storage type codes and their meanings.

**Code    Meaning**

- C    Character.
- F    Four byte, floating point.
- G    Eight byte, floating point (double precision).
- I    Two byte integer.
- L    Four byte integer.
- P    Packed decimal (BCD, or binary coded decimal).

This field is only relevant for field specifications with N or S edit codes.

This completes the definition of the field specifications that are used in the *product* file.

**2-16 Planning and Defining Your Application**

## HP TurboIMAGE/iX Field Specifications

For field specifications that you will use in HP TurboIMAGE/iX data set definitions, you can specify data manager specific attributes for the field specification at this point. If you press the **Data Mgr Attribs.** function key, the data manager specific field attributes screen is displayed. These attributes have default settings, so you don't need to complete them. Refer to the *HP ALLBASE/4GL Developer Reference Manual* if you are interested in finding out more about data manager specific field attributes.

If you pressed the **Data Mgr Attribs.** function key, press the **Base Fld Specs.** function key to return to the field specifications screen.

---

## Summary

In this lesson you learned the characteristics of field specifications. The field specifications stored in the dictionary define the essential characteristics of the fields used in your application.

You also learned how to use the dictionary field specifications screen to set up the fields for your application. The field specifications details that you entered are only definitions; physical file space will be allocated later.

For HP ALLBASE/SQL tables and KSAM files, the actual allocation of physical file space occurs when you create the files required by your application. You will do this in lesson 3.

Physical space for an HP TurboIMAGE/iX data set cannot be allocated from within HP ALLBASE/4GL. To allocate physical space for an HP TurboIMAGE/iX data set, you must use normal HP TurboIMAGE/iX database creation methods.

This completes the first of three tasks required to define the data used in the *training* application. Your next task is to define the record layouts of your files.

---

## Lesson 2 - Defining Record Layouts

---

### Objectives

When you complete this lesson, you will have learned how to:

- Define a record layout
- Use the record header screen to create a record header
- Use the record layout screen to create layout details
- Generate a record layout

### Record Layouts

A record layout defines the format of the records in a data file. A record can be associated with one or more files and is used in logic blocks, screens, and reports to qualify a field reference. The layout specifies the names of the fields and their sequence in the record. It also specifies which fields are used as key fields for accessing records in the file. When you create the record layout, HP ALLBASE/4GL defines a record buffer for reading and writing records.

### Data Manager Rules

The data manager you use will have certain rules about record layouts, which you will need to follow within HP ALLBASE/4GL. The KSAM file manager, for instance, requires at least one key to a record so it can maintain an index of the records in a data file.

The record layout for an HP ALLBASE/SQL table does not need to have any fields defined as key fields. If you do define a field on the record layout as a key field, HP ALLBASE/4GL uses the field to create an index for the table. You cannot use a record layout that contains field specifications with multiple occurrences.

HP TurboIMAGE/iX data set definitions require that record layouts:

- Record layouts must contain no key fields with repeated occurrences.
- Master data set layouts must contain no keys that allow duplicate entries.

### 2-18 Planning and Defining Your Application



- Automatic master data sets must contain one key field and no other fields.
- Manual master data sets must contain one key field and may contain other non-key fields.
- Detail data sets need not contain any key fields.

### **Task 1 - Creating a Record Layout Header**

To create a record layout you must first complete the record layout header screen and then complete the record layout details screen.

Header screens are used for many components of HP ALLBASE/4GL. Logic blocks, screens, and reports all have header screens. The information that you enter in a header screen is used as an index and is recorded in the index portion of the HP ALLBASE/4GL system files (S-files). Detail screen information is stored in the data portion of the HP ALLBASE/4GL S-files.

#### **Using the Record Layout Header Screen**

This screen enables you to describe a record for use in the application.

##### **To access the screen:**

1. From the main menu, select the *Dictionary* option
2. Choose *Record Layouts*
3. Choose *Header*.



### Record Layout Header Screen

The record layout header screen is shown as it will appear after you have made the entries for the product record header.

#### To enter the field values:

Field	Entry	Explanation
Record Name	product_rcrd	The name used to refer to the record in a file definition or file record field reference.
Secured	Accept the default.	Specifies whether this item is secured against modification by an unauthorized developer.
Description	Enter a description for documentation purposes.	

#### To complete and commit the screen:

1. When all of the values on the screen are correct, press the **Commit Data** function key to create the record layout header.

## 2-20 Planning and Defining Your Application

This completes the work required to define the values for the record layout header screen.

## **Task 2 - Defining Record Layout Details**

Now that you have defined the record layout header, you will define the details of the record layout.

1. Press the **Record Details** function key to go to the record layout details screen, or use the **Previous Menu** function key to go to the record layout details screen via the menus.

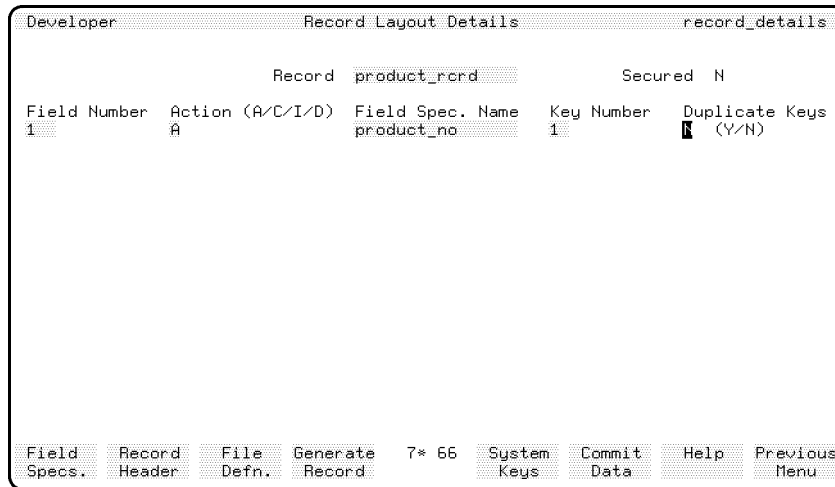
### **Using the Record Layout Details Screen**

This screen enables you to define the field specifications making up this particular record layout. You can also state whether a field is to be a key for the file, and if it is, whether duplicate values are permitted in the field.

As with the process of defining a field specification name, you are not allocating any physical space.

#### **To access the screen from the main menu:**

1. Select the *Dictionary* option.
2. Choose *Record Layouts*.
3. Choose *Details*.



### Record Layout Details Screen

The screen is shown as it should look when you have completed entering the values for the first field.

#### To enter the field values for the first field:

Use the table below to enter the field values depicted on the screen.

Field	Entry	Explanation
<b>Record</b>	Accept the default.	The name of the record layout. Processing can only continue if you have already completed the header for this record layout. Enter the name yourself if it is not displayed.
<b>Field Number</b>	Accept the default.	Defines the order of the fields contained in the record. As you define the structure of the record layout, you must declare each field in ascending field order. All the numbers must be contiguous, but it is possible to insert an extra field into the record layout at a later stage. This is the first field in the <i>product_rcrd</i> record.

## 2-22 Planning and Defining Your Application

Field	Entry	Explanation
<b>Action</b>	Accept the default.	Defines the action you wish to perform on this field. An entry of <i>A</i> indicates <i>Add</i> a new field.
<b>Field Spec. Name</b>	Enter <code>product_no</code>	Name of a previously defined field specification. If you try to enter the name of a field that you have not defined, HP ALLBASE/4GL displays an error message.  HP ALLBASE/4GL uses this entry to determine how much space to allocate for the data stored in this field, as well as the valid data types for the field.
<b>Key Number</b>	Enter <code>1</code>	Used for keyed file access purposes.
<b>Duplicate Keys</b>	Enter <code>N</code>	Indicates whether duplicate records with the same key value can occur on the one file. An entry of <i>N</i> indicates that a given key value may occur in only one record in the file.

#### To complete and commit the entries:

1. Press `Commit Data`. This displays a scroll area below the data entry line.

This scroll area shows the details of your entry. As you enter more fields into the record layout, this display reflects the current state of the record layout.

The display shows the field numbers, names, key numbers, and whether duplicate keys are permitted.

#### Completing the Record Layout Details

The entries shown below are required to complete the record layout details. Only the values that require an entry are shown below.

If you make an error in a field you have committed, you can rectify it easily. First, enter the field number of the line that you want to edit, then accept the action code of *C*, rectify the error and then press the `Commit Data` function key.

To complete the entries for KSAM and SQL based applications:

Field No. 2

Field	Entry
Field Number	Accept the default of 2
Action	Accept the default of A
Field Spec. Name	Enter <code>description</code>

Field No. 3

Field	Entry
Field Number	Accept the default of 3
Action	Accept the default of A
Field Spec. Name	Enter <code>supplier_no</code>
Key Number	Enter 2
Duplicate Keys	Enter Y

Field No. 4

Field	Entry
Field Number	Accept the default of 4
Action	Accept the default of A
Field Spec. Name	Enter <code>lead_time</code>

You will see that one of the fields above has a key number. In total, the product file has two keys. This allows you to read the file (or print reports) according to the product number or the supplier number. Only the product

## 2-24 Planning and Defining Your Application

number field is defined as a unique key since duplicate values may occur in the other fields.

The following screen image shows how the record layout screen should appear when you have entered all the field details.

Developer		Record Layout Details		record_details	
		Record	product_rord	Secured	N
Field Number	Action (A/C/I/D)	Field Spec. Name	Key Number	Duplicate Keys (Y/N)	
5	A				
1	product_no	1	N		
2	description				
3	supplier_no	2	Y		
4	lead_time				

Field Specs.	Record Header	File Defn.	Generate Record	7* 15	System Keys	Commit Data	Help	Previous Menu
--------------	---------------	------------	-----------------	-------	-------------	-------------	------	---------------

**Completed KSAM and SQL Record Layout Details Screen**

### Entry Notes

#### Action Codes

When defining items there are a number of occasions where you will be prompted for an action to be performed. The codes are universal, but some items use more actions than others. The codes are:

- *A* = Add
- *C* = Change
- *I* = Insert
- *D* = Delete

If an item does not exist, the default is *A*; if the item already exists, the default action is *C*.

### key fields

The key number defines the index number to be associated with this field. Every key number in a record must be unique. Key number 1 is called the primary key.

The KSAM file manager requires at least one key so it can maintain an index for the location of a record in a file. Without a key, no indexed access to a record via that field is possible. The key number has no relationship to the field number.

The HP TurboIMAGE/iX file manager requires one key for master data sets, but no keys are required for detail data sets.

For the HP ALLBASE/SQL data manager, no key fields are needed.

### To complete entries for HP TurboIMAGE/iX based applications:

#### Field No. 2

Field	Entry
Field Number	Accept the default of 2
Action	Accept the default of A
Field Spec. Name	Enter description

#### Field No. 3

Field	Entry
Field Number	Accept the default of 3
Action	Accept the default of A
Field Spec. Name	Enter supplier_no

## 2-26 Planning and Defining Your Application

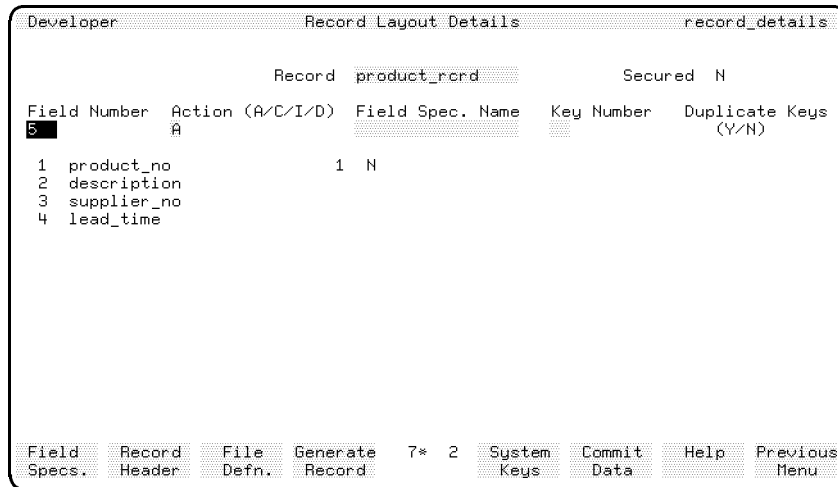


**Field No. 4**

<b>Field</b>	<b>Entry</b>
<b>Field Number</b>	Accept the default of 4
<b>Action</b>	Accept the default of A
<b>Field Spec. Name</b>	Enter lead_time

You will see that none of these fields have key numbers. A manual master data set may only contain one key field.

The following screen image shows how the record layout screen should appear when you have entered all the field details.



**Completed TurboIMAGE Record Layout Details Screen**

### Task 3 - Generating a Record Layout

Before a record layout can be used in an application, it must be generated.

Generating the record layout creates a run-time version of the record layout and performs additional validation and processing. The generation process resolves pointers to the field specifications, and produces an executable form of the record definition.

#### To generate a record layout:

1. After pressing **Commit Data** to commit the last set of fields, Press the **Generate Record** function key. This starts the generation process.

When you use the generate program for the first time, two messages are displayed. The first message tells you that *generate* is being called, and the second confirms that the record itself is being generated.

```
Developer          Record Layout Details          record_details
                                     Record product_rcrd          Secured N
Field Number  Action (A/C/I/D)  Field Spec. Name  Key Number  Duplicate Keys
5
1  product_no          1  N
2  description
3  supplier_no
4  lead_time

Generation successful.          (16001)

Field  Record  File  Generate  7*  2  System  Commit  Help  Previous
Specs.  Header  Defn.  Record   Keys  Data  Menu
```

#### Successful Generation

Generate is a large program and is a separate unit in the HP ALLBASE/4GL suite of programs. To avoid the need to call generate repeatedly, it remains resident in memory until you exit from the developer. The next time you use generate in this session, the *calling generate* message will not be displayed.

### 2-28 Planning and Defining Your Application

If you receive an error from the generate process, a generate error screen shows the number of the field that is in error, and the type of error encountered. Note these details and use them to assist you in correcting the record.

If the generate is successful, you will receive a *Generation successful* message indicating that the record layout has been correctly generated.

You have now created and generated a record layout. In the next lesson you will use the record layout to create a data file.

While developing an application you may want to look at the details of an existing record layout.

**To look at existing record layout details:**

1. Enter the record layout name in the *Record* field of the *Details* screen
2. Press **Return**.

The contents of the record layout will then be displayed.

---

## Summary

In this lesson you created a record layout.

To create the record layout you performed the following tasks:

1. You defined a record layout header.

This lets you name a record layout, and enter a description of it.

2. You defined the record layout details.

You specified the fields that make up the record layout, and the order of the fields in the record.

3. You generated the record.

The generation process calls an external program that validates the details of a record layout. Generation resolves references to the fields on the record, and creates an executable form of the record layout.

The final task in setting up the data fields is to define and create the files.

---

## Lesson 3—Defining Data Files

---

### Objectives

When you have finished this lesson, you will have defined and created a data file. Depending on the type of application you have chosen to create, this file may be a single KSAM data file, an HP ALLBASE/SQL table, or an HP TurboIMAGE/iX data set.

If you are developing the KSAM based application, continue reading below.

If you are developing the HP ALLBASE/SQL based application, turn to the next section.

If you are developing the HP TurboIMAGE/iX based application, turn to the section for TurboIMAGE/iX that follows in this lesson.

---

### KSAM Data File Definition and Creation

The data file you will create in this lesson is a KSAM file. KSAM is an abbreviation for “Keyed Sequential Access Method”. HP ALLBASE/4GL includes a KSAM data manager that allows you to create and access KSAM data files.

When you created the record layout in Lesson 2, you allocated a key number to several fields on the records layout. The KSAM file manager uses the key fields to define the indexes for the file. Key number 1 in a record layout is called the primary key.

#### 2-30 Planning and Defining Your Application

## Task 1 - Defining a Data File

Just as you had to define the field specifications before you could create the physical fields, you have to follow the same process for data files. Your first task is to define the characteristics of the file.

### Using the Data File/SQL Table Definition Screen

You use this screen to specify the name of the file to be used in HP ALLBASE/4GL, the name by which it is known to the host operating system, and the record layout to be used for the file. You can also enter a brief description for documentation purposes.

#### To access the screen:

1. From the main menu, select the *Dictionary* option.
2. Choose *Database Items*.
3. Choose *Data File/SQL Table Definition*.

The screenshot shows a window titled "File/SQL Table Definition" with a menu bar containing "Developer", "File/SQL Table Definition", and "file\_defn". The main area contains the following fields and values:

File Name	product	File Type	E (I/S/F/U/A/M/D)		
Description	product				
AUTHOR:	vtodd				
Last Modification:	Date	3/19/92	Time	13:20:05	
External Name	PRODUCT				
SQL DBEfileset	PRODFS				
SQL Access Class	1 (1 - Public, 2 - Publicread, 3 - Private)				
Record Layout	product_rcrd				

At the bottom, there is a menu bar with the following options: Field Specs., Record Header, SQL Sel. Details, Create File, 3\* 56, System Keys, Commit Data, Help, and Previous Menu.

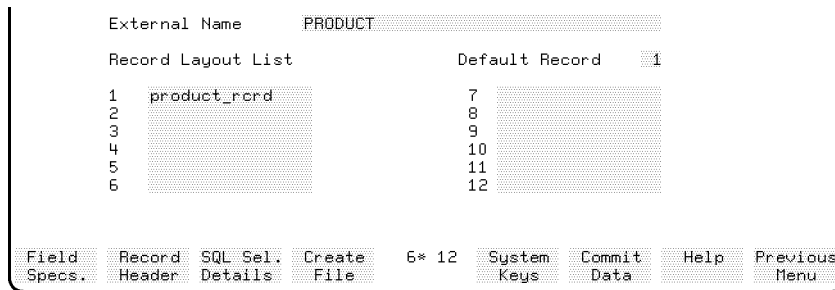
**Data File/SQL Table Definition Screen Showing Window**

This screen uses three different windows. One window allows you to define KSAM and serial data files. The second window allows you to define

HP ALLBASE/SQL tables. The third window allows you to define HP TurboIMAGE/iX data sets. Your entry in the *File Type* field determines which window is displayed.

**To enter the field values:**

Field	Entry	Explanation
<b>File Name</b>	product	Internal name of the data file used by the application.
<b>File Type</b>	I	Indicates the type of file interface to be used. Automatically displays KSAM window.
<b>Description.</b>	Enter a description.	



**KSAM Window**

**To enter values in the KSAM window:**

Field	Entry	Explanation
<b>External Name</b>	product (Accept default, if shown)	Name of the file in the MPE/iX environment. By default this is the same name as the internal file name. A different name can be entered.

**2-32 Planning and Defining Your Application**

Field	Entry	Explanation
<b>Default Record</b>	Accept the default.	Indicates the record layout number to be used if no record name is specified in a file field reference. Also used to create the file.
<b>Record Layout List</b>	This is the header for the next 12 fields. You can enter the names of up to 12 different record layouts to be associated with this file.	HP ALLBASE/4GL always builds the physical disk file records according to the default record layout.
<b>Record Layout List - 1</b>	<code>product_rcrd</code>	Name of the record layout to be associated with this file.
<b>Record Layout List - 2-12</b>	Leave these fields blank.	

### Notes on Window Entries

In addition to defining KSAM files, the other options for the *File Type* field allow you to define SQL base tables (S), fixed length record serial files (F), variable length record serial files (V), HP TurboIMAGE/iX automatic master data sets (A), HP TurboIMAGE/iX manual master sets (M), and HP TurboIMAGE/iX detail data sets (D). Refer to the *HP ALLBASE/4GL Developer Reference Manual* for more information about these file types.

Using multiple record layouts in the file definition allocates a file buffer in memory (when the application is running) for each record layout assigned to the file. If you use multiple record layouts for a file, you can read more than one physical file record at the same time, or you can read the same physical record through different file buffers, allowing you to modify one buffer without destroying the contents of another.

### To complete and commit the screen:

1. Press the `Commit Data` function key to create the file definition.

This is the logical file definition for use in HP ALLBASE/4GL.

2. Create the physical disk file. by pressing the **Create File** function key, or use the menus by pressing the **Previous Menu** function key and selecting *Create File/SQL Table*.

The file/SQL table creation screen will be displayed.

You are now ready to create the data file.

## **Task 2 - Creating the Data File**

This is the second step in creating a data file. During this procedure the KSAM file manager is automatically invoked to create the physical files that are necessary to support the data file that you have defined.

### **Using the File/SQL Table Screen**

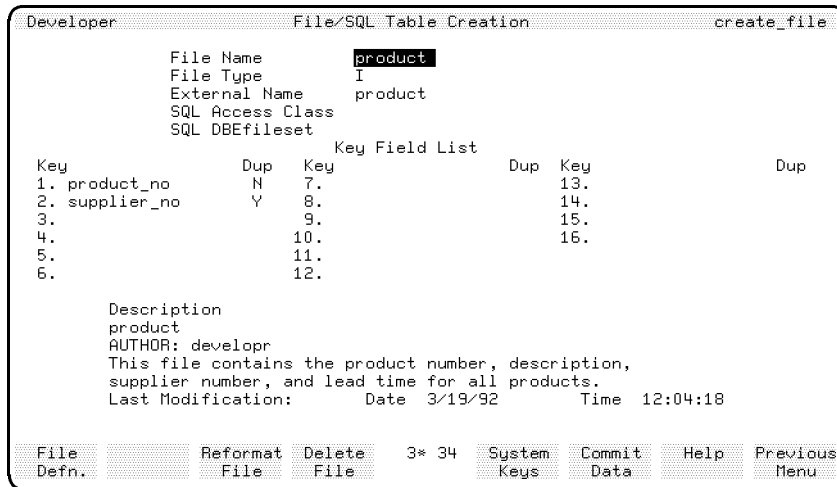
On this screen you enter the name of the data file that you defined in the last lesson. HP ALLBASE/4GL reads the file definition and displays the details so you can confirm your selections. You then create the physical file space.

#### **To access the screen from the main menu:**

1. Select the *Dictionary* option.
2. Choose *Database Items*.
3. Choose *Create File/SQL Table*.

## **2-34 Planning and Defining Your Application**





**Create File/SQL Table Screen**

The screen is shown as it will appear after you have completed all of the entries.

**To enter the field values:**

Field	Entry	Explanation
File Name	Accept the default of <b>product</b>	The name you used when you created the file definition.

This completes the data entry required on this screen. The remaining fields display the data you entered on the record layout screen.

**To create the file on disk:**

1. Press **(Return)** to display the details of this file on the screen.

The MPE/iX external file name is displayed first, followed by the key fields from the default record.

Note that the two keys defined when you created the record definition are listed, along with an indicator specifying whether duplicate values are

allowed. The description of the file is displayed in the lower portion of the screen.

2. Press the **Commit Data** function key to create the file on disk.

When this process has completed, HP ALLBASE/4GL displays a message indicating that the file has been successfully created.

---

## Summary

In this lesson you created the KSAM data file for your application.

To create the file you completed the file definition, then used the file creation screen to build the physical disk files for the data file.

The KSAM file manager uses the key fields on the default record for the file to define an index for the location of each record in the data file. Key number 1 in a record layout is called the primary key.

## What To Do Next

You have now finished defining the fields, the record layout, and the file required for this part of the application. The rest of this chapter contains instructions for using other data managers. Check those over, then turn to the Self-Test Questions for this chapter.

---

## HP ALLBASE/SQL Table Definition and Creation

When you have completed this lesson you will have learned the two steps necessary for creating an HP ALLBASE/SQL table:

- Completing the definition for a base table.
- Creating the table.

---

### Note



If you intend to work through this section, make sure your HP ALLBASE/4GL administrator has already created an HP ALLBASE/SQL database that corresponds with the field specifications, record layouts, and file definitions that you have created, and those that you will create in later lessons.

If that has not been done, you can still read through this section, view the screens, and make the field entries. You will not, however, be able to commit the screen.

---

### Task 1 - Defining an HP ALLBASE/SQL Table

HP ALLBASE/4GL allows you to store data for an application in HP ALLBASE/SQL tables. You have already defined the fields that make up the table records. You have also defined the record layout that is to be associated with the data file.

When you created the record layout in Lesson 2, you allocated a key number to several fields on the records layout. The HP ALLBASE/SQL data manager uses the key fields to define the indexes for the file. Key number 1 is the primary key.

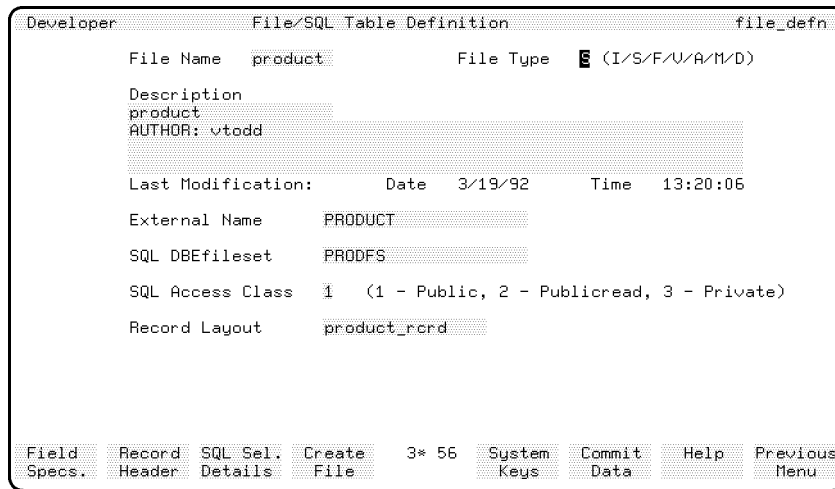
#### Using the Data File/SQL Table Definition Screen

This screen uses three different windows. For an SQL table, this screen allows you to specify the name of the table as it is known to HP ALLBASE/4GL and the record layout to be used for the table.

This screen also allows you to specify the external name of the table, the SQL DBEFileSet to hold the table, the SQL access class for the table, and the name of the record layout for the table.

**To access the screen from the main menu:**

1. Select the *Dictionary* option.
2. Choose *Database Items*.
3. Choose *Data File/SQL Table Definition*.



**Data File/SQL Table Definition Screen**

**To enter the field values:**

Field	Entry	Explanation
<b>File Name</b>	product	The HP ALLBASE/4GL name for the SQL table. This is the name you use in references to the table in HP ALLBASE/4GL logic. Note that the name can only be up to eight characters long.
<b>File Type</b>	S	Indicates the type of file interface that is to be used in this application. <i>S</i> specifies an HP ALLBASE/SQL table. Automatically displays window.

**2-38 Planning and Defining Your Application**

Field	Entry	Explanation
Description	Enter text describing table's purpose	

```

External Name      PRODUCT
SQL DBEfileset    PRODFS
SQL Access Class  1  (1 - Public, 2 - Publicread, 3 - Private)
Record Layout     product_rcrd

Field  Record:  SQL Sel.  Create  6* 12  System:  Commit  Help  Previous
Specs.  Header:  Details  File                               Keys   Data   Menu

```

### SQL Window

#### To enter values in the SQL window:

When you entered S in the *File Type* field and pressed **(Return)**, the SQL window appeared. Fill in that window with the values given below.

Field	Entry	Explanation
External Name	Accept the default.	The name of the table in the HP ALLBASE/SQL database environment for the application.
SQL DBEfileset	PRODFS	The name of the DBEfileset for the table. The DBEfileset must exist in the database environment before you can create the file. The entry in this field defaults to <i>SYSTEM</i> , but you can type over the default.
SQL Access Class	Accept the default.	Specifies the table locking mode as <i>public</i> , <i>publicread</i> , or <i>private</i> . All transactions on this table are subject to the normal HP ALLBASE/SQL locking provision.
Record Layout	product_rcrd	

**To complete and commit the screen:**

1. When you have entered all the values for this screen, press the **Commit Data** function key to create the file definition.

This is the logical file definition for use in HP ALLBASE/4GL. Your next step is to create the physical disk file.

2. Press the **Create File** function key, or use the **Previous Menu** function key and select *Create File/SQL Table*.

The file/SQL creation screen should be displayed.

**Task 2 - Creating an HP ALLBASE/SQL Table**

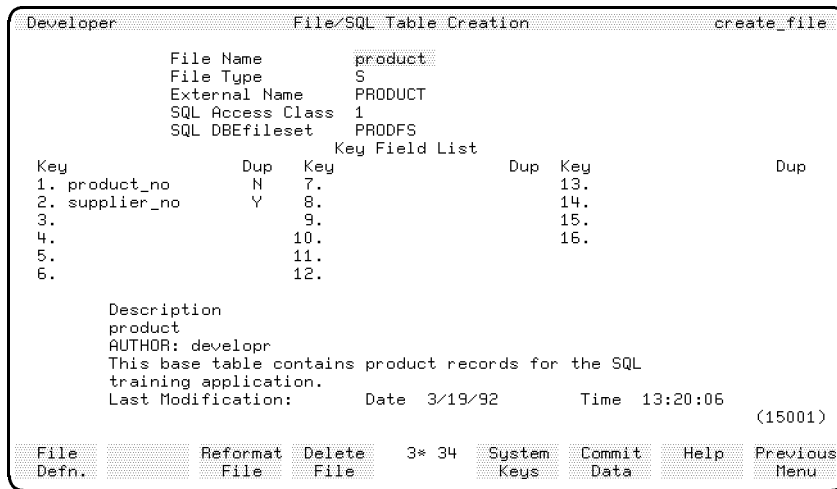
This is the second step in creating an HP ALLBASE/SQL table. During this procedure, HP ALLBASE/4GL creates the table, using the field specifications in the table record layout to define the columns of the table.

**Using the Create File/SQL Table Screen**

You use this screen to enter the name of the table you have already defined. HP ALLBASE/4GL reads the table definition and displays the details so you can confirm your selections. You can then create the table.

**To access the screen:**

1. From the main menu select the *Dictionary* option.
2. Choose *Database Items*.
3. Choose *Create File/SQL Table*.



**Create File/SQL Table Screen**

**To enter the field values:**

Field	Entry	Explanation
<b>File Name</b>	Accept the default.	The name you used when you created the table definition.

This completes the data entry required on this screen. The remaining fields display the data you entered on the record layout screen and the file/SQL table definition screen.

### To create the physical file:

1. Press **Return** to display the details of this file on the screen.

The MPE/iX external file name is displayed first, followed by the key fields from the default record.

Note that the two keys, defined when you created the record definition, are listed, along with an indicator specifying whether duplicate values are allowed. The description of the file is displayed in the lower portion of the screen.

2. To create the table, press the **Commit Data** function key.

HP ALLBASE/4GL displays a message telling you that it is connecting to the application database, and then displays a message to confirm that the table has been created successfully.

### Notes on SQL Tables

#### SQL Table Ownership

HP ALLBASE/4GL creates tables as *owner\_group.table* where *owner\_group* is the SQL owner group defined for the application. The name of this owner group is specified on the application definition screen in the administrator application.

The *product* table is created as *SQLGRP.PRODUCT* in the application database environment.

#### Table Format

HP ALLBASE/4GL creates the table using the field specifications in the table record layout to define the columns of the table. HP ALLBASE/4GL creates columns defined by numeric field specifications as decimal columns, and columns defined by other field specification types as CHAR columns. All columns created permit null values.

HP ALLBASE/4GL uses the key fields in the record layout for a table to create indexes for the table. The index corresponding to key field number 1 is a clustering index.

## 2-42 Planning and Defining Your Application



---

## **Summary**

In this lesson you created the definition for an SQL table, and then created a base table in the application database environment.

## **What To Do Next**

The rest of this chapter describes how to define an HP TurboIMAGE/iX data set.

When you are ready, turn to the Self Test Questions to check your understanding of the material in these lessons.

---

## HP TurboIMAGE/iX Data Set Definition

This lesson shows how to define the HP TurboIMAGE/iX data set for this part of the application.

HP ALLBASE/4GL can store data for an application in HP TurboIMAGE/iX data sets. You have already defined the field specifications that will correspond to HP TurboIMAGE/iX data items. The record layout you defined is the HP ALLBASE/4GL equivalent of a data entry.

As well as defining the HP TurboIMAGE/iX data set within HP ALLBASE/4GL, a physical data set in an HP TurboIMAGE/iX database must be created. Usually, the database would be created first, and the field specifications, record layouts and file definitions would be uploaded to HP ALLBASE/4GL using an HP ALLBASE/4GL utility. In this situation you would not need to redefine these items within HP ALLBASE/4GL. Refer to the HP TurboIMAGE/iX Interface section in the *HP ALLBASE/4GL Developer Reference Manual* for details of the upload utility.

---

### Note



If you intend to work through this section, make sure your HP ALLBASE/4GL administrator has already created an HP TurboIMAGE/iX database that corresponds with the field specifications, record layouts, and file definitions that you have created, and those that you will create in later lessons.

If that has not been done, you can still read through this section, view the screens, and make the field entries. However, you will not be able to commit the screen.

---

The database created by your administrator should also contain the automatic master data sets that are normally linked to each key field in each data set.

### Task 1 - Defining an HP TurboIMAGE/iX Data Set

As with the KSAM and SQL data sets, HP TurboIMAGE/iX data sets are defined on the file definition screen.

## Using the File Definition Screen

On this screen you specify the name of the database that will contain the data set, the name by which the data set is known to the database system, and the record layouts to be used for the data set. You also enter a brief description that is used for documentation purposes.

### To access the screen from the main menu:

1. Select the *Dictionary* option.
2. Choose *Database Items*.
3. Choose *Data File/SQL Table Definition*.

```
Developer      File/SQL Table Definition      file_defn
File Name  product      File Type  M (I/S/F/U/A/M/D)
Description
Last Modification:      Date  3/26/92      Time  13:57:17
Field Specs.  Record Header  SQL Sel. Details  Create File  3* 56  System Keys  Commit Data  Help  Previous Menu
```

### Data File/SQL Table Definition Screen

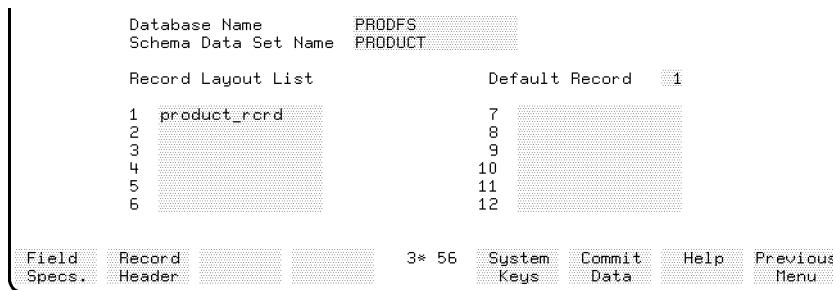
The third window for this screen, selected by entering M in the *File Type* field, allows you to define HP TurboIMAGE/iX data sets.

### To enter the field values:

Although you cannot commit the screen unless you have a TurboIMAGE database created, you can make the entries listed below to view the screen and window shown here.

Field	Entry	Explanation
File Name	product	Internal name of the file used by the application. Note that the name can be no longer than eight characters.
File Type	M	Indicates the type of file and interface to be used. In this case it is a manual master data set in the HP TurboIMAGE/iX file interface.
Description	Enter a file description.	

HP ALLBASE/4GL displays the HP TurboIMAGE/iX file window automatically.



**HP TurboIMAGE/iX Window**

## 2-46 Planning and Defining Your Application

**To enter values in the TurboIMAGE/iX window:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Database Name</b>	<code>traindb</code>	<p>Name of the HP TurboIMAGE/iX database that contains the data set you are defining. Can be up to eight characters long.</p> <p>This is not the external name of the HP TurboIMAGE/iX database, but rather, an internal name defined by the HP ALLBASE/4GL administrator in the <i>administ</i> application.</p>
<b>Schema Data Set Name</b>	Accept the default.	<p>Name of the data set as it is known in the external HP TurboIMAGE/iX database schema file.</p> <p>The name may contain all characters that are acceptable in HP TurboIMAGE/iX data sets. All lowercase alphabetic characters are converted to uppercase characters.</p>
<b>Default Record</b>	Accept the default.	<p>Indicates the record layout number to be used if no record name is specified in a file field reference. It is also used by the HP TurboIMAGE/iX data manager to create the file.</p> <p>You can't use a value other than 1 if you only have one record layout defined for the data set in the record layout list.</p>

Field	Entry	Explanation
<b>Record Layout List</b>	This is the header for the next 12 fields. You can enter the names of up to 12 different record layouts to be associated with this file.	HP ALLBASE/4GL always builds the physical disk file records according to the default record layout.
<b>Record Layout List - 1</b>	product_rcrd	Name of the record layout to be associated with this file.
<b>Record Layout List - 2-12</b>	Leave these fields blank.	

### Notes on Windows

For information about the three File/SQL Table Creation windows and multiple record layouts, refer to the Field Notes section for KSAM files.

### To complete and commit the screen:

1. Press the **Commit Data** function key to create the file definition.

If you do not have an HP TurboIMAGE/iX database defined, you will get an error message.

When completed, this is the logical file definition for use in HP ALLBASE/4GL.

---

## Summary

This lesson showed how to define the HP TurboIMAGE/iX data set for your application.

The HP TurboIMAGE/iX data manager uses the key fields on the default record for the file to define an index for the location of each record in the data set. Key number 1 in a record layout is called the primary key.

### 2-48 Planning and Defining Your Application

---

## Self Test Questions

You have now finished defining the fields, record layouts, and files required for this part of the application.

Check your understanding of these lessons by answering the Self Test Questions below.

**Question 1.** Which of the following are valid HP ALLBASE/4GL names?

- cust-rec
- ltest
- example\_1
- prod.rec
- administratorname

**Question 2.** HP ALLBASE/4GL names can be from 1 to 16 characters long. Which of the following are exceptions to this rule?

- File names.
- Screen names.
- SQL select lists.
- Action items.

**Question 3.** Do *N-number* and *N-NUMBER* refer to the same numeric constant?

**Question 4.** Where are HP ALLBASE/4GL field specification names defined?

- Screen painter.
- Dictionary.
- Record layouts.
- Screen details.

**Question 5.** What are record layouts used for?

**Question 6.** Explain the difference between file definition and file creation?

**Question 7.** Which of the following items do you need to generate?

- Field specifications.
- Files.
- Record layouts.

**Question 8.** If a field specification has a U edit code, which of the following values would be accepted in this field?

- FROG
- fish
- 34
- alpha1
- BETA1



---

## Answers

**Answer 1.** example\_1 and ltest are valid HP ALLBASE/4GL names.

HP ALLBASE/4GL names cannot contain hyphens or periods, and cannot be longer than 16 characters long.

**Answer 2.** File names and SQL select list names cannot be up to 16 characters long. Both of these can have a maximum of only 8 characters.

**Answer 3.** No, N-number and N-NUMBER refer to different numeric constants. HP ALLBASE/4GL is case sensitive.

**Answer 4.** Field specifications are defined in the Dictionary.

**Answer 5.** Record layouts are used in file definition. The record layout defines the format of records in a file. It specifies the fields, their sequence, and which fields are to be used as key fields.

**Answer 6.** File definition defines the characteristics of a file, such as its external name, its HP ALLBASE/4GL name, and the record layouts associated with the file.

File creation physically allocates space to, and creates, a file.

**Answer 7.** Record layouts need to be generated. Files and field specifications don't need to be generated.

**Answer 8.** All of them.

When you enter a lower case value in a field defined as type U edit code, HP ALLBASE/4GL automatically upshifts the letters. Numbers are accepted too.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Developing Screens

---

Phase three of developing an application involves creating the screens you will use. The two lessons in this chapter show you how to create the menu and the data screen for the *training* application.

You will learn how these HP ALLBASE/4GL screens are used to develop an application:

- Screen Header Screen
- Screen Field Details Screen
- Screen Painter

The screen painter screen is part of an HP ALLBASE/4GL function called the “screen painter.” This function lets you determine the positioning of various fields on your screens. You can use the screen painter to create or modify screens for your application.

To help you understand the process involved in creating the screens for an application, the following section describes some of the screen fundamentals for HP ALLBASE/4GL.

---

## How HP ALLBASE/4GL Screens Work

HP ALLBASE/4GL screens use quite different screen handling techniques than you may have used in more conventional programming environments. In particular, HP ALLBASE/4GL screens provide extensive facilities for automatic data formatting, data validation, and data movement.

All of the developer screens you use in developing the *training* application are in fact HP ALLBASE/4GL screens. They demonstrate the features you can include in your own applications.

HP ALLBASE/4GL screens and the automatic logic associated with them form a fundamental part of any application you develop. When HP ALLBASE/4GL displays a screen, the screen and logic associated with the screen control the application. In particular, screens control all interaction between your application and the user. The type of screen, and the way the screen is called, determines where control resumes after the screen processing is complete.

### Screen Types

HP ALLBASE/4GL uses three types of screens. They are:

- Menus.
- Data screens.
- Windows.

#### Menus

Menus have the following features:

- They present the user with a range of options in the application. You have already been using menus to select options in the developer application.

Frequently, an option on a menu calls a further menu. For example, this occurs when you select the *Dictionary* option on the developer main menu.

- They allow the use of the keyboard to select options. You have already used the keyboard facilities to select options from the developer menus. Exactly the same keyboard facilities are available to the end users of your applications.

### 3-2 Developing Screens

- No action occurs until the user selects and activates an option.

When HP ALLBASE/4GL displays a menu, all current activity in the application (except background processing) is terminated immediately. Nothing further happens until the user selects and executes one of the options on the menu, or initiates an action from a function key.

In general, HP ALLBASE/4GL displays a menu when an application first starts and when the current process terminates.

### **Data Screens**

Data screens allow the user to enter data for the application and allow the application to display information for the user. Data screens contain input fields, display fields, and literals or titles. Data screens can also have a scroll area for information display. This is particularly useful if you have more information to display than can readily fit on the screen.

Data screens have the following properties:

- They form the basis of all interactive data input and information display for the application.
- They are the primary means of user data entry.
- They are the primary means of information display.
- They can usually perform most of the data formatting, data validation, and data movement required by an application.

### **Windows**

A data screen can be overlaid with a window screen. A window screen can present additional information to the user as well as accepting input from the user.

A window uses the same types of screen items and can perform the same data manipulation as a data screen. However, HP ALLBASE/4GL can only display a window on an already displayed data screen. Menus cannot use windows.

You have already seen and used a window on the file/SQL table definition screen in the dictionary menu. You may want to go back to this screen now to refresh your memory.

## **Function Keys**

All screens can use a set of function keys. The function keys allow the user to select various application options or system functions that may be needed in the application. Besides the default function keys available in HP ALLBASE/4GL, you can set up function keys that are specific to your application.

## **System Items**

All screens can contain system items. These are fields that display the contents of various system or application defined fields. System items can display the following information for the end user:

- The current date and time.
- Communication area fields.
- Master titles.
- Application titles.
- Numeric or alphanumeric constants.
- Variables or calculated items.
- Scratch-pad fields.

## **Displaying Screens**

HP ALLBASE/4GL displays a screen when it receives a SCREEN command in logic, or a D- action from a menu, a function key, or a decision table.

The SCREEN command is one of the logic commands that can be called in a function or a process. After you have learned how to create the screens for your application, you will learn how to use HP ALLBASE/4GL to create a process that contains a SCREEN command.

Once HP ALLBASE/4GL displays a screen, the screen itself controls the sequence of events occurring in the application. In general, the screen remains in control until the user initiates an action that terminates the screen. This occurs regardless of the type of screen.

### **3-4 Developing Screens**

For a menu, no further actions or processing occurs until the user selects and activates a menu option.

For a data screen, the screen remains active until the user initiates an appropriate action to terminate the screen, such as the “Commit Data” action.

In general, interaction with the user is not possible if HP ALLBASE/4GL is not displaying a screen.

---

## Lesson 4 - Creating Menus

---

### Objectives

When you have finished this lesson, you will have created the main menu for the *training* application.

You will have learned how to:

- Use the screen header screen to define the name and the main characteristics for the main menu.
- Use the screen painter to enter, edit, move, and delete text on the screen, to create system items, and to create a menu action item and the text for the item.

---

### Developing a Menu

Defining a menu involves two tasks: defining the screen header screen and creating the actual menu image.

The menu you are about to create initiates a process that controls the *product\_scrn* data entry screen. You will define the process itself in lesson 6.

### 3-6 Developing Screens



---

## Task 1 - Creating the the Menu Header

The *Screen Header* screen is used for menus, data screens and window screens. When you specify that you are defining a menu, certain fields on the screen header screen default to entries that are appropriate for menus.

### Menu Path

**To access the screen from the main menu:**

1. Select the *Screens* option.
2. Choose *Header*.

```
Developer: Screen Header screen_header:
Screen main Secured (Y/N)
Screen Type M (M/D/W)
Automatic Numbering Y (Y/N)
Clear Fields on Commit B (I/D/B/N)
Function Keys main
Screen Help Name main

Scroll: Top line Bottom Line Direction (U/D)

Window Starting Line Number

Description
main
AUTHOR: developr
This is the main menu for the training application.

Last Modification: Date 3/16/92 Time 15:14:24

Function Keys Screen Painter Field Details Generate Screen 3* 58 System Keys Commit Data Help Previous Menu
```

**Screen Header Screen**

**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Screen</b>	<b>main</b>	Name of the menu to be used in this application; name used to select this menu using the menu bypass feature.
<b>Secured</b>	Accept the default.	Specifies whether item is secured against changes by an unauthorized developer.
<b>Screen Type</b>	<b>M</b>	Indicates type of screen being defined. <i>M</i> indicates that the screen is a menu.
<b>Automatic Numbering</b>	No entry	Default entry cannot be modified when defining a menu.
<b>Clear Fields on Commit</b>	No entry	Default entry cannot be modified when defining a menu.
<b>Function Keys</b>	Accept the default.	
<b>Screen Help Name</b>	Accept the default.	These fields are covered in more detail in a later lesson.
<b>Scroll and Window Description</b>	No entry Enter a description.	Fields automatically bypassed for menus.

**To complete the entries:**

1. When you have completed the entries, press the **Commit Data** function key to create the screen header.

The screen header is now complete. You can now paint the screen image associated with this header using the screen painter.

2. Return to the previous menu and select *Painter* from the screen development menu.

### **3-8 Developing Screens**

---

## Task 2 - Painting the Menu Details

To paint the menu details, you use the screen painter program. The screen painter allows you to interactively create the menu image that is seen by the end user.

The screen painter can be used to create all types of screens. HP ALLBASE/4GL varies some of the available functions depending on what type of screen you are developing.

A menu is composed of literal text, system items such as the date and time, and menu items. A menu item is composed of two parts; its prompt (what the user sees) and its action (what is executed when the user activates the item).

When you use the screen painter, you build an exact image of the screen as it appears in the application. You use the cursor keys to position items on the screen and the function keys to select the various available screen painter options. Any screen item can be located anywhere on the screen.

The screen painter also provides facilities that allow you to modify screens by copying, moving, and deleting single items or groups of items on a screen.

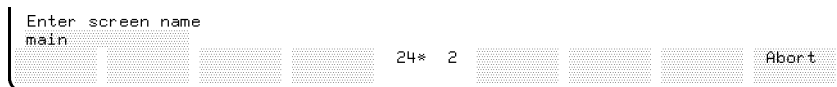
### Menu Path

1. From the main menu, select the *Screens* option.
2. Choose *Painter*.

The screen painter screen will be displayed.

### Using the Screen Painter to Paint a Menu

When you enter the screen painter for the first time, the screen is cleared and the following prompt appears at the base of the screen.



**Screen Painter Prompt**

### To use the screen painter:

1. Enter the name of the screen you want to paint.

The last screen name used in any of the screen definition screens appears as a default entry. In this example the name should default to *main*.

2. Press **Return** to start the screen painter.

---

#### Note



If you want to abort the screen painter at any stage you can press the **Abort Painter** function key, which is accessed through the **More Keys** function key. The screen painter asks you to confirm the request. Enter **Y** in response to this message to abort the screen painter. Any input you have entered will not be saved.

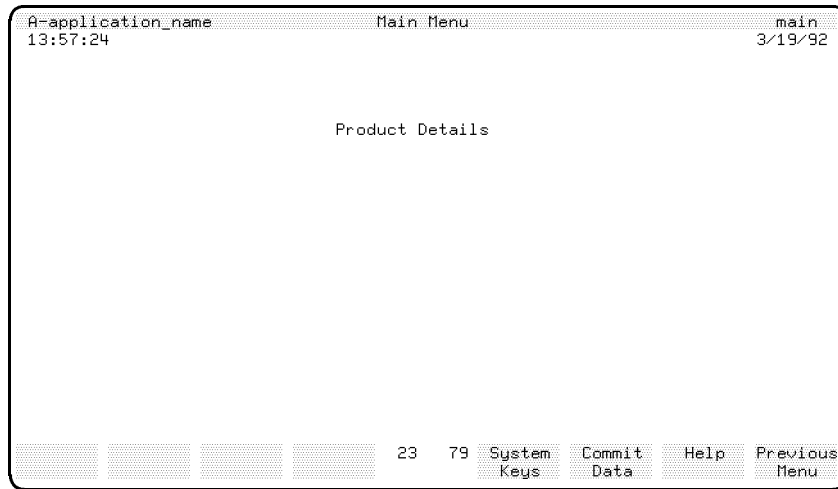
---

When you enter the screen name, the painter positions the cursor at the top left corner of the screen, displays some screen identification details at the bottom of the screen, and displays the screen painter function keys.

On most terminals, the numbers corresponding to the current cursor position are displayed between the two sets of four function key labels. The number on the left is the line number, and the number on the right is the column number. This information helps you position your screen items exactly where you need them.

The next figure shows the screen image you are about to paint.

## 3-10 Developing Screens



### Menu for Training Application

#### Step 1 - Creating the Screen Title

The first item you need to enter on this menu is the screen title. It will be located on the top line of the screen.

The top line of a screen is known as the “banner line”. When the screen is displayed at run-time, the banner line is displayed in half-bright inverse video. The system administrator can change the display highlighting of all screen items.

#### To enter the screen title text:

1. Use the cursor keys, **cursor right**, **cursor left**, **cursor up** and **cursor down** to move the cursor to line 1, column 36 on the screen.

The cursor coordinates are displayed between the two sets of function key labels at the bottom of the screen.

2. Press **Return** when you have moved the cursor to the correct position. The painter displays a message telling you that it is in *Enter Text* mode.
3. Type in the text **Main Menu**.
4. Terminate the text item and the *Enter text* mode by pressing **Return**.

This paints the menu heading. To complete the screen, you will enter some system items and define actions to be performed.

Before continuing, read the following paragraphs for useful tips on how to perform actions quickly in the painter such as moving, copying, and deleting text and blocks of text.

## Some Useful Tips on Working with the Painter

### Using the Cursor

When in text entry mode, you can use the **(Insert char)**, **(Delete char)**, **(Clear line)** and **(Back space)** keys to edit the text you are entering. Try it out. Type in a few extra items of text anywhere on the screen. Remember to press **(Return)** to start text entry mode, and to press **(Return)** again to terminate each entry. Don't worry about cluttering up the screen with extra entries because you will soon learn how to delete them.

Notice that as you move onto a defined item it is highlighted and it becomes the current item. When an item is highlighted, you can modify it by pressing **(Return)** and typing characters over the item. The screen painter enters the *Modify text* mode if you press **(Return)** while an item is highlighted. Press the **(Return)** key to complete a text entry or an alteration.

### Using the Tab Key

After you've typed in a few text items, try pressing the **(Tab)** key. Each time you press it, the next item on the screen is highlighted until you get to the last item on the screen.

Pressing the **(Tab)** key again returns the highlight to the first item on the screen. Pressing **(Shift) + (Tab)** followed by **(Return)** takes you backwards through the fields.

Notice that as you move from one field to another using this method, the cursor is positioned on the first character of the current item.

Now try pressing the **(cursor home)** key followed by **(Return)**. The cursor is positioned in the upper left corner of the screen. When you press **(Shift)** and **(cursor home)** followed by **(Return)**, the cursor is positioned on the last item on the screen.

## 3-12 Developing Screens

If you've tried out a few of the items described above, your screen is probably a scattered mess of text items. If you don't have more than one text item on your screen, enter some now as you'll need them for the next exercise in this lesson.

### Using Layout Keys

The screen painter layout keys are a set of function keys allowing you to move, copy, or delete items on a screen.

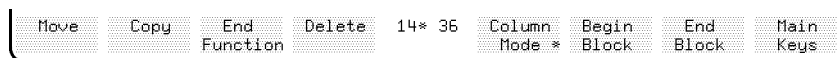
#### To use the layout keys:

1. Make sure you are at the main level of function keys.

You can tell this by the fact that one of the function keys is labelled **Exit**. If it isn't, press the **Main Keys** function key.

2. Press the **Layout Keys** function key and you will see a new set of keys displayed.

The *Layout Keys* allow you to modify the layout of existing items on the screens.



#### Screen Painter Layout Keys

By using the layout keys, you can move, copy and delete single items or blocks of items on the screen. For now, all you want to do is delete the text fields that you don't need on this screen.

#### To delete items on the screen:

1. Use the **Tab** key or cursor keys to select the first field you want to delete, and then press the **Delete** function key.

The screen painter asks you to confirm the deletion.

2. Type **Y** and then press **Return** to confirm the deletion.

The item is erased from the screen. This is how you can delete a single item from a screen.

**To practice deletions:**

1. Select another item that you want to delete.

So you can delete a group of fields, choose one that has a few more items below or to the right of it. If you don't have an area that looks like this, type in a few extra text items so it does.

2. Press the **Delete** function key.

The screen painter displays the confirmation message again.

3. Press the **Delete** function key again.

The next field on the screen is highlighted as well.

As you keep pressing the **Delete** key, the highlight keeps advancing until you reach the last field on the screen. Each time you press the **Delete** function key, the confirmation message is displayed. If you keep pressing the **Delete** function key (and no other), you will stay in *delete mode*.

4. When you have highlighted the last field you wish to delete, type **Y** and then press **Return** to confirm the deletion.

All of the highlighted fields are deleted in one operation.

The same method also works for moving and copying screen items. If you highlight a number of fields by repeatedly pressing a function key, the action will be performed on all of the highlighted fields.

When you're moving or copying items, the painter issues a warning message if you specify a destination that causes a clash of fields or do not allow sufficient room on the screen for any of the items to be moved or copied. If this happens, move the cursor to a new destination and then attempt the move or copy again.

### **3-14 Developing Screens**



## Using the Block Function

The screen painter provides another way of performing a layout function with a group of screen items. This is a two-step block function. First, you define a block of fields, then you choose a function to perform on the defined block.

### To define a block:

1. Move the cursor to one corner of the rectangle which is to become the defined block, and then press the **Begin Block** function key.
2. Move the cursor to the diagonally opposite corner of the block and press the **End Block** function key.

All the fields that are partially or wholly within the block are highlighted and the function you choose is performed on all of them.

You are **always** asked to confirm your actions when you are using the block delete function. This helps ensure that you don't accidentally destroy part of a screen that you are creating or modifying.

### To perform different functions on a block:

1. Define a block of fields on your screen, and then press the **Move**, **copy**, or **delete** function keys.

When you're asked to move the cursor to the new position, move the cursor to the new position of the upper left corner of the block and then press the **Return** key. The fields within the block are moved or copied to the new location, retaining their relative positions within the block, or they are deleted.

If any of the fields to be moved or copied clash with any other fields already on the screen, the painter issues a warning message and the move or copy is not allowed.

2. To turn the *copy* mode or the *move* mode off, or to abort the *delete* mode, press the **End Function** function key.

Create a few text items and try moving or copying single fields about the screen. Try pressing the **Move** or **Copy** key successively to use a group of

items in *move* or *copy* mode. When you feel comfortable with these keys, move on to the next part of the lesson.

---

**Note**

The exercises you have completed so far have only used text items. All field types, system items, action items, text, and input or display fields behave exactly the same way. As well as moving the images of fields on the screen, the screen painter also moves or copies the attributes of the fields.

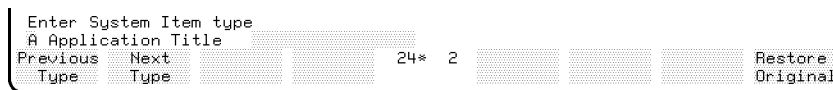
---

Clean up the screen so you have only the text item **Main Menu** located on the first line at column 36. Then return to the main screen painter function keys (by pressing the **Main Keys** function key if necessary), so that you can complete painting this menu.

## Step 2 - Defining System Items for a Menu Screen

System items include the date and time, the current screen name, and some dictionary defined items.

You create these system item fields on the screen by using the *System Item* prompt, which is accessed with the **System Item** function key.



### System Item Prompt

Some of the available system items are:

- Communication areas.
- The date.
- Application titles.

#### To display the screen name:

You can use a system item to display the name of the current screen.

## 3-16 Developing Screens

1. Move the cursor to column 76 on the first line of the screen and follow these steps to create the screen name item:

Action	Explanation
Press <b>System Item</b>	This displays the system item prompt at the base of the screen.
Press <b>Next Type</b>	This displays another system item. Pressing the function key again specifies and displays the next item type until you reach the end of the list. The list then starts again.
Press <b>Next Type</b> until <b>*</b> Communication Area appears	This is an abbreviation for <i>Communication Area Field</i> . You will learn more about these fields later but for now it is sufficient to know that they are values maintained automatically.
Press <b>Return</b>	This moves the cursor from the small single character prompt in the left field to the larger field on the right.
Type SCREEN	This is a reference to the communication area field *SCREEN, which contains the name of the current screen.
Press <b>Return</b>	This creates the system item field on the screen. The prompt is cleared and the current screen name, along with the cursor, is displayed at line 1, column 76.

#### To display the date:

You can also use a system item to display the current system date on a screen.

1. Move the cursor to line 2, column 72 and follow the next steps to create the system maintained date item.

Action	Explanation
Press <b>System Item</b>	This displays the system item prompt at the base of the screen.
Press <b>Previous Type</b>	This displays another system item type. Pressing this function key specifies the previous type until you reach the start of the list of available types. You then start stepping through the list from the end again.
Specify <b>D Date</b>	This is the system date field. It is maintained by the system. The current system date is displayed whenever this screen is used by the application.
Press <b>Return</b>	This creates the system item date field on the screen. The date, along with the cursor, is displayed at line 2, column 72. Note that the date is displayed in either US or European format. The date format on your system is set by the system administrator.

### To display the time:

A communication area field called \*TIME holds the current system time. You can use a system item to display this field on a screen.

1. Move the cursor to line 2, column 2.
2. Follow the steps below to create a field that displays the current time.

## 3-18 Developing Screens

Action	Explanation
Press <b>System Item</b>	This displays the system item prompt at the base of the screen.
Specify <b>*</b> Communication Area	*TIME is a communication area field.
Press <b>(Return)</b>	
Type <b>TIME</b>	This refers to the *TIME communication area field.
Press <b>(Return)</b>	This creates the time field on the screen.

#### To display an application title:

An application title is a literal string that can appear anywhere on a screen. Application titles are stored in the dictionary, and any changes you make to an application title are automatically reflected in all screens that use the title. A typical use for application titles is to display a corporate or department name on all screens in an application. In this example, you will use an application title to display the name of the application on the screen.

1. Move the cursor to line 1, column 2.
2. Follow the steps below to create a field that will display an application title.

You will create the application title in a later lesson.

Action	Explanation
Press <b>System Item</b>	This displays the system item prompt at the base of the screen.
Specify <b>A</b> Application Title	This is another type of system item.

Action	Explanation
Press <b>Return</b>	
Enter <code>application_name</code>	<p>The painter displays a warning message informing you that this item hasn't been defined in the dictionary. You can still create the system item on the screen, creating a reference to the named <i>application title</i> even though the item hasn't been defined yet.</p> <p>At this stage you will only see <i>A-application_name</i> displayed. The screen painter does not need to have this field defined when you first refer to it. When you define the field later, it is displayed automatically in this area.</p>

### Step 3 - Defining Actions for the Menu Screen

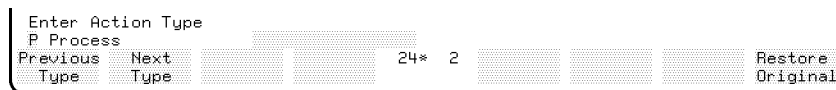
So far you have only entered a text item and some system items. On a menu you must also define at least one action to be performed when the user selects the associated prompt.

A menu action can execute any of a number of items in HP ALLBASE/4GL. These include processes, background processes, reports, or other screens.

The following steps allow you to create an action which, when selected, executes an HP ALLBASE/4GL process.

#### To define actions to be performed:

1. Position the cursor at line 7, column 32.
2. Press **Action**. This displays the action item prompt at the base of the screen.



**Action Item Prompt**

## 3-20 Developing Screens

3. Complete the following steps to create the action item.

Action	Explanation
Press <b>Next Type</b>	Pressing this key specifies and displays the next action type until you reach the end of the list of available action items. You then start stepping through the list again.
Specify <b>P Process</b>	This indicates that this item executes a process when it is selected. You will create the process itself in a later lesson.
Press <b>(Return)</b>	This moves the cursor from the small single character prompt on the left field to the larger field to the right. The system is now waiting for you to enter the name of the process associated with this item on the menu.
Type <b>product_proc</b>	This is the name of the process that is executed when the user chooses the <i>Product Details</i> selection from the menu.
Press <b>(Return)</b>	This creates the name of the action item on the screen. The process name, along with the cursor, is displayed at line 7, column 32. Now you can overwrite the name of the action with the prompt that is seen by the end user.
Enter <b>Product Details</b>	This action automatically invokes the text editing mode and overtypes the process name that formed the original action title.

You can edit the text of an action title just like any text field. However, the highlighting differs from that of a text field to enable you to distinguish between the two.

An end user can select the various choices on a menu by typing the first letter of an item and then pressing **(Return)**. If you use unique initial letters for each of the items on any one menu, an experienced end user can move rapidly through menus.

This completes the steps for creating a menu. Your next task is to save the menu.

---

## Task 3 - Saving the Menu Screen

The menu screen does not need to be generated. Exiting from the screen painter will automatically save the menu input.

### To exit from the screen:

1. Press the **Exit** function key to exit from the screen painter.

The screen painter automatically saves the screen under the name that you allocated. In this case, the name is *main*. The screen is now ready to be used in the application.

---

## Summary

In this lesson you defined the main menu for the *training* application.

Menus allow users to control the application by selecting the item they want. HP ALLBASE/4GL can execute the following types of actions as a result of user selections from a menu:

- A further menu or a data screen.
- A function or a process.
- A help screen.
- A background process.
- A report.
- An external program.

Creating menus requires two steps:

1. Use the screen header screen to define the menu header.

### 3-22 Developing Screens



This defines details such as the menu name, and the names of the function key set and help screen associated with the menu. The screen header also allows you to enter a description of the menu.

2. Use the screen painter to create the menu image.

The screen painter allows you to define text literals, action items and system items on a menu.

Menus are automatically saved when you exit from the screen painter. Menus don't need to be generated.

---

## Lesson 5 - Creating Data Screens

---

### Objectives

When you have completed this lesson, you will have learned how to create a data screen for the *training* application.

You will have learned how to perform the following four tasks:

- Define the screen header using the screen header screen.
- Paint the screen image using the screen painter. This includes performing these steps:
  - Painting system items and text items on a data screen.
  - Using “column mode” to enter vertically aligned screen items.
  - Creating a number of data input fields on the screen.
- Define the screen field details using the screen fields details screen. This task includes performing these steps:
  - Specifying the data movement details for the input and display fields on a data screen.
  - Specifying the behavioral characteristics of screen fields.
  - Associating a logic function with a screen field.
  - Specifying the data validation and data formatting performed for each screen field.
- Generate the data screen.

### 3-24 Developing Screens

---

## Defining Data Screens

Data screens let the user enter data and allow the application to display information to the user. Data screens can consist of a combination of literals, system items, input fields, and display fields.

The data screen you create in this lesson will let the user enter the information necessary to add, delete, modify, or review records in the product data file, table, or data set.

---

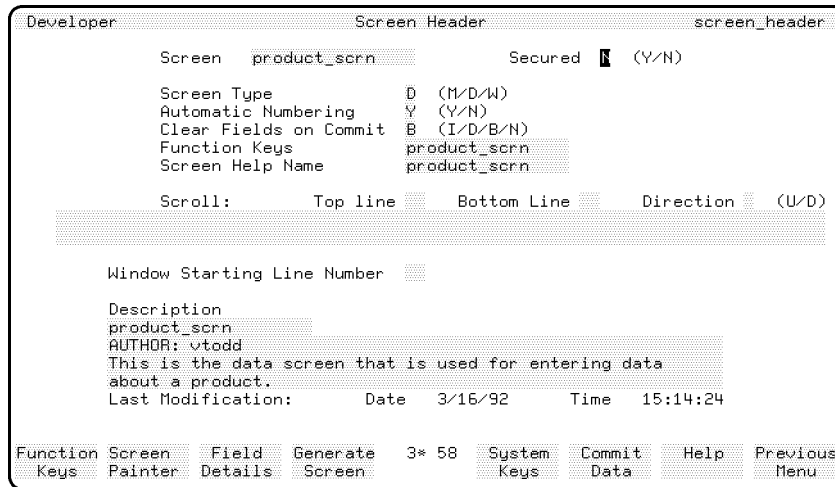
### Task 1 - Defining the Screen Header

#### Menu Path

1. From the main menu, select the *Screens* option.
2. Choose *Header*.

#### Description

This screen allows you to define the essential characteristics of a data screen. It's the same screen you used to define the header for the menu.



### Data Screen Header Screen

To enter the field values:

Field	Entry	Explanation
Screen	product_scrn	The name of the data screen in this application.
Secured	Accept the default.	
Screen Type	D	Indicates the screen type you are defining. <i>D</i> indicates a data screen.
Automatic Numbering	Accept the default.	Field specifies the way in which the data fields on the screen are numbered. <i>Y</i> specifies automatic numbering.  This means that the fields are numbered sequentially from left to right, and top to bottom, starting at the top leftmost field. This determines the order in which the fields are processed as the user follows the standard tabbing sequence through the screen.

### 3-26 Developing Screens

Field	Entry	Explanation
<b>Clear Fields on Commit</b>	Accept the default value of B.	Indicates which fields are cleared when the user presses the <b>Commit Data</b> function key. (See Entry Notes section.)
<b>Function Keys</b>	Accept the default.	
<b>Screen Help Name</b>	Accept the default.	
<b>Scroll: Top line</b>	Don't enter anything in this field.	
<b>Window Starting Line Number</b>	Don't enter anything in this field.	The <i>Function Keys</i> , <i>Screen Help</i> , <i>Top Line</i> and <i>Window Starting Line Number</i> fields are described in detail in later lessons.
<b>Description</b>	Enter a description for documentation.	

### Entry Notes

Possible entries for the *Clear Fields on Commit* field are:

I	=	Input.
D	=	Display.
B	=	Both.
N	=	None.

### To complete and commit the screen:

1. When you have completed all of the field entries, press the **Commit Data** function key to create the screen header.

The screen header is now complete, and you can create the screen image with the screen painter.

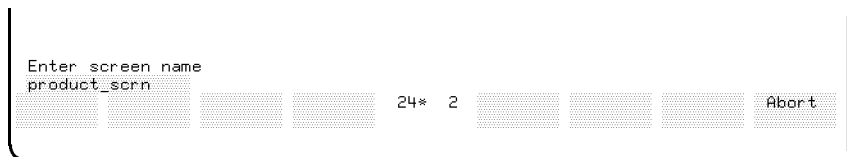
---

## Task 2 - Painting the Data Screen Image

### To run the screen painter:

1. Press the **Screen Painter** function key  
or press the **Previous Menu** function key.
2. Select the *Painter* option.

When the screen painter starts, HP ALLBASE/4GL clears the screen and displays the following prompt at the base of the screen.



**Screen Painter Prompt**

### Step 1 - Entering Text Items

Check the name of the screen entered here.

If you have just defined the *product\_scrn* data screen in the screen header screen, the default name of *product\_scrn* should appear. If not, then enter the name now.

Here is the screen you are going to paint:

```
Training Application      Product Details      product_scrn
14:38:10                3/19/92

      Product Number  [ ]

      Description      [ ]

      Supplier Number  [ ]

      Lead Time        [ ]

[ ] [ ] [ ] [ ] 14* 35 System Keys  Commit Data  Help  Previous Menu
```

### Product Details Screen

#### To enter system items:

First create the following system items, in the same way that you did in the previous lesson.

1. Display the application title `application_name` at the top of the screen at line 1, column 2.
2. Enter the screen heading `Product Details` as a text literal at line 1, column 35.
3. At column 68 of the same line, display the communication area field `*SCREEN`.
4. At line 2, column 2 display the communication area field `*TIME`.
5. At column 72 of the same line, display the `DATE` system item to complete the system items required on this screen.

This completes the definition of the header lines for this screen.

## Tips On Entering Text

To save yourself time in creating the lines for this screen, read the following information on using column mode.

### Using Vertically Aligned Prompts

This data screen has a number of vertically aligned prompts. Each of them has a data input field to the right of the prompt.

During the creation of text items on the menu that you have already defined, the cursor remains at the end of a defined item when you press the **(Return)** key.

When you are entering a number of vertically aligned text items, you can use the screen painter *column mode*. In column mode, the cursor returns to the beginning of the text item, one line down, when you press **(Return)** to terminate an entry. Then you can immediately enter the next prompt, vertically aligned with the previous.

### To use column mode:

1. To set the column mode on, press the **Layout Keys** function key and then press the **Column Mode** function key.

An asterisk appears in the function key label indicating that column mode is now on.

2. To turn column mode off, simply press the **Column Mode** function key again.

The asterisk is cleared from the label when you turn the column mode off.

## Step 2 - Entering the Screen Prompts

### To enter vertically aligned items:

1. Set the column mode on and position the cursor at line 6, column 18.
2. Press **(Return)** and type in **Product Number**.
3. Press the **(Return)** key to complete the entry.

The cursor will be directly under the “P” in the word “Product”.

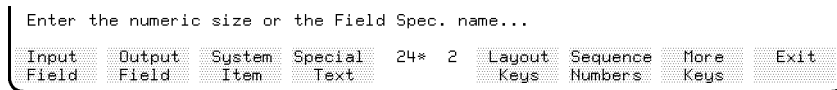
## 3-30 Developing Screens



4. Press **cursor down** until the cursor is on line 10.
  5. Press **Return** and then enter **Description**.
  6. Enter **Supplier Number** at line 12.
  7. Enter **Lead Time** at line 14.
  8. Turn column mode off and press the **Main Keys** function key.
- You can now define the data fields for this screen.

### Step 3 - Defining Data Fields

To complete this screen you must define the input fields and display fields on the screen image. This is done with two function keys at the *Main Keys* function key level: the **Input Field** function key, and the **Output Field** function key.



#### Input Field and Output Field Function Keys

Two different methods are available for defining input and display fields. You can retrieve the details of a field specification from the dictionary by entering its name, or you can indicate the length of the field now and complete the other details later. Both are described here.

#### To use the Dictionary Field Specification:

1. Move the cursor to line 6, column 35 and follow these steps to create an input field for the product number, using the dictionary field specification.

<b>Action</b>	<b>Explanation</b>
Press <b>Input Field</b>	The input field prompt appears at the bottom of the screen.
Type <b>product_no</b>	The painter creates an input field at the base of the screen as you type the name. This is the name of an existing dictionary field specification. If the field specification does not exist, the field creation is aborted.
Press <b>Return</b>	This terminates the entry of the field specification name. A field of the correct length is displayed at line 6, column 35.

By using the dictionary field specification name, you have linked the dictionary definition of the field to this particular screen field. When you define further options for some of the field details, you will see that the painter has automatically retrieved the dictionary field specifications for this field.

2. Use the same procedure to create the description field. Move the cursor to line 10, column 35 and follow these steps.

<b>Action</b>	<b>Explanation</b>
Press <b>Input Field</b>	This displays the input field prompt at the bottom of the screen.
Enter <b>description</b>	This creates the description field on the screen.

#### **To specify the number of characters:**

You can also create an input field by specifying the number of characters for the field.

1. Move the cursor to line 12, column 35 and follow these steps to create the input field for the supplier number.

### **3-32 Developing Screens**

Action	Explanation
Press <b>Input Field</b>	This displays the input field prompt at the bottom of the screen.
Type 6	This is the size of this field in characters.
Press <b>Return</b>	This displays a 6 character field.

The field has been created with no reference to a dictionary defined field specification. You will enter the details later in the screen field details screen.

2. To create the lead time input field, move the cursor to line 14, column 35 and follow these steps.

Action	Explanation
Press <b>Input Field</b>	This displays the input field prompt at the bottom of the screen.
Enter 2	This completes the creation of the field.

As with the supplier number field above, no reference has been made to a dictionary field specification. You will do this later using the screen field details screen.

#### To create display only fields:

You can create display only fields using the painter in the same way you create input fields. The only difference is that you use the **Output Field** function key instead of the **Input Field** function key.

You will define some display only fields in one of the later lessons.

#### Step 4 - Saving the Screen

The screen is now complete and you can save it.

1. Press the **Exit** function key, and the screen is saved automatically. Control then passes back to the level from which you called the painter.

You can now proceed to define details for the fields on this data screen. This includes the details omitted because you didn't use the field specification name method to define some fields, as well as other details you will learn about in the next section. If you are at the screen development menu, select *Details*. If you are at the screen header screen, press the **Field Details** function key.

---

### **Task 3 - Specifying Screen Field Details**

After you have created the image of a data screen with the screen painter, you usually need to specify some further details for the data fields on the screen.

These details include automatic data movement, field-associated functions, and some special field processing indicators. For two fields on your data screen, some default values will already be created based on the field specification you used to define the field. In most of these cases you will only need to add the automatic data movement specifications.

#### **Menu Path**

##### **To access the screen field details screen from the main menu:**

1. Select the *Screens* option.
2. Choose *Details*.

#### **Screen Description**

This screen allows you to complete the definition of each input or display field on an application data screen.

Developer		Screen Field Details		screen_fields	
Screen	product_scrn	Sequence Number	1	Secured	N
DATA MOVEMENT	Field Spec. Name	product_no			
Primary	F=product_no.product	Data<-->Screen			
Default		Data -->Screen			
Other 1		Data<-- Screen			
2		"			
3		"			
Input/Display	I (I/D)	Required Field	Y (Y/N)		
Help name	product_no	Clear on COMMIT	N (Y/N)		
Include in SHOW	Y (Y/N)	Automatic Flow to Next	N (Y/N)		
Function	product_key_read	Prior/After/Both	A (P/A/B)		
Perform On SHOW	N (Y/N)	After Function if Blank	I (Y/N)		
Edit Code	U (X/A/U/K/N/S/Q/D/T)	Justification	L (L/R/C/N)		
Field Length	6 Min 6 Max	Decimal Places			
Range		Pad Character			
Table		Blank When Zero	(Y/N)		
Screen Header	Screen Painter	Function Keys	Generate Screen	17* 63	System Keys
					Commit Data
					Help Previous Menu

### Screen Field Details Screen

First you must enter the name of the screen, and then step through the fields on the application data screen completing their respective details.

The complete screen field details screen is used for each field on your application screen.

#### To enter the field values:

Field	Entry	Explanation
Screen	product_scrn	The name of the data screen that you are defining field details for.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Sequence Number</b>	Accept the default.	<p>The field number, defined by the sequence of data fields on your data screen.</p> <p>The field sequence for this screen is set automatically from top left to bottom right. This sequence of fields and allocation of field numbers is determined by the <i>Automatic Numbering</i> flag on the screen header screen.</p> <p>When you press the <b>Return</b> key, the details already recorded for this field are displayed at the bottom of the screen. The dictionary specification details have been automatically transferred to this screen, and many of the fields contain a default value.</p>
<b>Field Spec. Name</b>	Accept the default.	<p>This field defaults to the dictionary field specification name used to define the attributes of this screen field in the screen painter.</p> <p>Blanking out this field changes the last nine fields on the field details screen to input fields. These fields cannot be changed when a link to the dictionary exists through a field specification name. If you have blanked out the field specification name, re-enter the name <b>product_no</b> to protect the screen field specification again.</p>
<b>Primary</b>	F-product_no.product	<p>This entry specifies that when the cursor is moved to this screen field, the current contents of the field <i>product_no</i> from the file buffer <i>product</i> are moved to the screen buffer and then displayed.</p> <p>On completion of data entry, the new contents of the screen field are moved to the file buffer. (Refer to “Entry Notes” section in this lesson.)</p>

### 3-36 Developing Screens

**To enter behavioral characteristics fields:**

The next six fields define the way the field behaves on the screen. This is not dependent upon any data type, format, or validation specifications.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Input/Display</b>	Accept the default.	This indicates whether this data field is an input field (I), or a display field (D). It enables you to override the setting you specified in the screen painter.
<b>Required Field</b>	Accept the default.	Indicates whether this field must be non-blank for alphanumeric, or non-zero for numeric fields, for the field value to be accepted.  When the user attempts to move off a required field that has been left blank, the field is highlighted and the cursor remains on it. The user can move back to a field with a lower sequence number, or enter data into the current field. The user cannot commit the screen with the <b>Commit Data</b> function key while a required field remains blank.
<b>Help Name</b>	Accept the default.	This is the name of the help screen that is called if the user presses the <b>Help</b> function key while the cursor is on this field.  If you have defined a help screen name for this field specification in the dictionary, that name is automatically transferred to this field. If you haven't specified a help screen name, HP ALLBASE/4GL creates a default help screen name with the same name as the field. You can change the help screen name here without affecting the dictionary specification.

Field	Entry	Explanation
<b>Clear on COMMIT</b>	Accept the default.	This field works in conjunction with the <i>Clear Fields on COMMIT</i> setting specified on the screen header for this screen. The relationship is an OR relationship. That is, if the setting for either is Y, the field is cleared. Only if both are set to N will the field not be cleared.
<b>Include in SHOW</b>	Accept the default.	This specifies whether the field should be manipulated when it is referred to from a logic block SHOW command. With some fields, you may want to ensure that the only time anything is ever done to them is when an explicit reference is made to them.  The SHOW command is used in HP ALLBASE/4GL logic to redisplay data in a screen field. You will learn about the SHOW command in a later lesson.
<b>Automatic Flow to Next</b>	Accept the default.	So far, all the fields that you have used require you to press the <b>Return</b> key to signal that the field entry is complete. In the same way, the fields you define for an application screen normally require the user to press the <b>Return</b> key to complete a field entry. Under some operating systems, setting the <i>Auto Flow to Next</i> field to Y causes HP ALLBASE/4GL to generate an implicit <b>Return</b> when the user fills the field. Although displayed on this screen, that facility is not available under the MPE/iX operating system.

### 3-38 Developing Screens



**To enter function specification fields:**

You can associate a logic function with each screen field. The next four fields specify how the function for this field is executed. This function enables you to perform some data manipulation or validation when data is displayed in the field, or entered into the field.

The field function you are naming here will read the product file for a record with a key value matching the value entered by the user.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Function</b>	<code>product_key_read</code>	This is the name of the function to be associated with this field. If you leave this field blank, entry into the next three fields is not possible.
<b>Prior/After/Both</b>	<code>A</code>	This entry specifies when the function is executed.  There are three available options. <i>P</i> indicates that the function is executed <i>prior</i> to data entry, when the user moves the cursor onto the field. <i>A</i> indicates that the function is executed <i>after</i> the completion of data entry for the field; this is only valid for input fields. <i>B</i> indicates that the function is executed both prior to and after data entry.

Field	Entry	Explanation
<b>Perform Function on SHOW</b>	Accept the default.	<p>This field relates to the behavior of the field under a SHOW command from a logic block.</p> <p>At times the function may perform some data formatting or extraction that is essential for the correct data to be displayed. In this case, the function needs to be executed when the field is displayed under control of a SHOW command.</p>
<b>After Function if Blank</b>	Accept the default.	<p>This field only applies to non-required screen fields. It indicates whether the function is executed even if the screen field has been left blank.</p> <p>You may have noticed that only one function can be called from the field, yet the function can be called at three distinctly different times; prior, after, or on SHOW. HP ALLBASE/4GL has two switches (*ENTERED and *SHOWING) that you can test in the function to determine when the function is being executed. This allows you to use the one function to perform three different operations, depending on the processing stage at which the function is executed.</p>

#### **To enter validation and format specification fields:**

The bottom nine fields on the screen are a copy of the same fields from the dictionary field specification screen.

If you have entered a field specification name in the *Field Spec. Name* field at the top of this screen, you cannot change any of this group of fields. You can only enter local field definitions if there is no field specification name associated with the application screen field.

The fields are not described here as they have exactly the same meaning as their counterparts in the field specification screen.

### **3-40 Developing Screens**

**To complete the Product Number screen:**

1. Once you have completed all of the details for this particular screen field, press the **Commit Data** function key.

When you press this key, the cursor returns to the *Sequence Number* field.

The details for the next field on the application screen are then displayed, enabling you to enter the remaining details.

As you go through the screen fields, HP ALLBASE/4GL creates a primary data movement identifier automatically. This data movement identifier consists of the field specification name for the current field and the file record specified for the first field on the screen.

**To complete the remaining screen field details:**

Now you are ready to complete the screen field details for the rest of the *training* screen.

- Enter the details listed. Only those fields that require an entry other than the default are listed here.
- After entering the details for each field, press the **Commit Data** function key.
- If you make an error in a particular field of the screen, you can display its details by entering its sequence number and pressing the **Return** key.

The field details are then displayed so you can modify them, and commit them again.

The following entries are required to complete the screen field details.

**Description**

Sequence Number	2
Primary	F-description.product

## Supplier Number

Sequence Number	3
Field Spec. Name	supplier_no
Primary	F-supplier_no.product

When you defined the supplier number field in the screen painter, you didn't use the dictionary field specification name method. For this reason, a default name is not displayed with the rest of the field details. When you enter the name *supplier\_no*, the fields at the base of the screen become display fields.

## Lead Time

Sequence Number	4
Field Spec. Name	lead_time
Primary	F-lead_time.product
Required Field	N

As with the previous field, you didn't use the dictionary field specification name method to define this field in the screen painter; therefore, no default name was displayed.

You have now completed the definition of the fields for this screen. As with a record layout, you must generate the screen before it can be used. The "Entry Notes" section that follows provides additional information for some of the fields. If you wish, skip this section and continue on to "Task 4 - Generating the Data Screen."

## Entry Notes

### Data Movement

The *Primary*, *Default* and *Other* fields define the automatic data movement that occurs as the user moves through the fields on the application screen. The three types of data movement are:

**Primary**                      When HP ALLBASE/4GL displays a screen field, it moves the contents of the *primary* data movement field

## 3-42 Developing Screens

to the screen and displays the field contents. For an input field, HP ALLBASE/4GL moves the contents of the screen field to the primary data movement field after the user completes data entry for the screen field.

**Default**

If the screen field is blank, HP ALLBASE/4GL moves the contents of the *default* field to the screen and then displays its contents. If the screen field is not blank, no action takes place.

**Other**

For input fields, HP ALLBASE/4GL moves the contents of the screen field to the *other* data movement fields when the user completes data entry for the field. Each input field can have up to three *other* data movement fields.

---

**Note**



You cannot specify both a primary and a default data movement field. They are mutually exclusive. A field can only have one source for its data, however, you can combine either of these “source” movement fields with up to three *Other* data movement specifiers.

---

The fields you name in these automatic movement specifiers can be any of the following:

- A file record field reference.
- A screen field reference.
- A scratch-pad field reference.
- A variable.
- A numeric or alphanumeric constant (default data movement field only).

---

**Note**

The file record field specifier consists of three components. The prefix *F*- indicates that this is a file buffer field reference. Next, the field name *product\_no* indicates the field to be used, and the file name *product* specifies the file buffer to be used. These last two terms are separated by a single period (. ). Refer to the *HP ALLBASE/4GL Developer Reference Manual* for further information on naming and referencing rules.

---

As the *Primary* and *Default* move specifiers are mutually exclusive (you can only move data into the screen field from one of these sources), HP ALLBASE/4GL skips over the *Default* move field. If you wanted to alter this field, you would need to delete the *Primary* move specifier, and then specify the *Default* move field. No *Other* move fields are to be entered in this example. When you leave the first *Other* move field blank, HP ALLBASE/4GL skips over the next two.

---

## Task 4 - Generating the Data Screen

A data screen must be generated to produce an executable version of the screen.

1. To generate the screen, press the **Generate Screen** function key on the screen field details screen.

You will see a message verifying that the screen is being generated.

As with record layout generation, the generate program validates field names and resolves pointers. If you have incorrectly specified an item on the field details, an error screen indicates where the error lies.

2. If you have made an error, go to the particular screen field, rectify it, and generate the screen again.

Once the screen is successfully generated, you can continue on to the next task.

You have now completed this lesson.

### 3-44 Developing Screens

---

## Summary

In this lesson you learned how to create a data screen.

Data screens allow the user to enter data and allow the application to display information to the user. Data screens can have both input and display fields, and can also use system items and text literals.

Data screens allow you to specify the following:

- Screen field data validation characteristics.
- Data movement associated with screen fields.
- Screen field behavior.
- Logic functions associated with fields.

Creating a data screen involves four tasks:

1. Defining the name and characteristics of the data screen on the screen header screen.
2. Painting the screen image with the screen painter. This involves painting system items, input fields, display fields, and text items on the screen.
3. Defining the screen field details. This involves specifying the data validation, data movement details, and behavioral characteristics for the input and display fields on the data screen. You can also associate logic functions with screen fields.
4. Generating the screen.

The data screen you created in this lesson allows the user to add, delete, modify, or review records stored in the *product* data file. In the next chapter, you will learn how to produce the logic to perform some of these tasks.

FINAL TRIM SIZE : 7.0 in x 8.5 in



# 4

## Defining Application Logic

---

### Logic

So far you have defined a number of field specifications and used them to create a data file and a data screen. The basic data requirements for this part of the application are now complete. Now you are ready for Phase 4 - defining the logic required to make the application run.

HP ALLBASE/4GL uses logic constructs called processes, functions, and SQL logic blocks. The lessons in this chapter describe the techniques to develop a process and a function so you can get your application running. If you are developing the HP ALLBASE/SQL based application, you will also develop an SQL logic block.

The screens you will learn how to use include:

- Process Header Screen
- Process Details Screen
- Function Header Screen
- Function Details Screen
- SQL Logic Block Header Screen
- SQL Logic Block Details Screen

### What Are Processes and Functions?

A process is similar to a program in a conventional language. When a process starts, HP ALLBASE/4GL initializes the environment to a known state. It closes all data files, clears all data buffers, and terminates any current non-background process, screen, function, report, or decision table.

A function is similar to a conventional subroutine. A function can be called from many places without automatically affecting the current environment. When a function is complete, the application returns to the place from which the function was called.

Processes and functions are collectively referred to as logic blocks. They both share the same constructs and generally the same commands. They each have a maximum of 99 lines of commands, and are created by entering the details of each line into formatted windows on HP ALLBASE/4GL screens. Using windows means that you don't need to remember the exact syntax for every command.

### **What Are SQL Logic Blocks?**

SQL logic blocks are the means of communicating with HP ALLBASE/SQL tables. SQL logic blocks are called from processes and functions to perform specific HP ALLBASE/SQL commands. For example, you may use the HP ALLBASE/SQL SELECT command to declare and open a cursor on an HP ALLBASE/SQL table.

An SQL logic block can contain up to eight SQL commands. At run-time, the commands in an SQL logic block are invoked by a command in an HP ALLBASE/4GL logic block.

---

## **Defining the Logic for the Training Application**

The logic blocks you are about to define specify how the application displays the *product\_scrn* screen and how the data entered on the screen is handled.

For the KSAM and HP TurboIMAGE/iX based applications, the logic blocks allow you to add data to the product file, or modify existing data in the file.

For the HP ALLBASE/SQL based application, the logic blocks allow you to add data to the product file, or view existing data in the file.

The logic blocks for the *training* application are:

- A process, called from the menu, to control the screen and the manipulation of the data file.

### **4-2 Defining Application Logic**

- A function, called from a data screen, to check for the existence of the record requested by the user.
- For HP ALLBASE/SQL applications, an SQL logic block to declare and open a cursor in the HP ALLBASE/SQL table, and an SQL logic block to commit data.

The logic blocks have been kept as simple as possible so you can get a basic file manipulation system running quickly.

---

## **Lesson 6 - Developing a Process**

---

### **Objectives**

When you have completed this lesson, you will have learned how to create a basic HP ALLBASE/4GL process.

You will have performed the three tasks necessary to create the process:

1. Define the process header using the process header screen.
2. Define the process details using the process details screen.
3. Generate the process.

---

## The Product Process

The process you are about to develop calls a data screen. When the user completes the screen, the same process writes the data entered by the user to the product file. The process then loops back to the beginning to start again.

Many processes call data screens. The *training* application uses a menu item to call a process, which in turn displays a data screen and then manipulates data in a file.

For the *training* application, the user must press the **Previous Menu** function key to exit from the process. After pressing the **Previous Menu** function key the user is taken back to the menu that called the process.

### Task 1 - Defining the Process Header

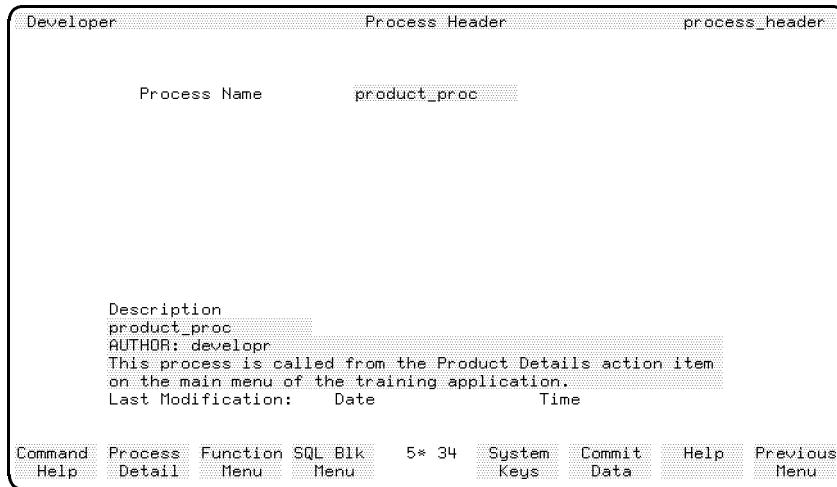
The first task in defining the process is to define the process header. This involves two steps. First you must define a process header, and then define the steps that make up the process on the process details screen.

#### Menu Path

##### To access the screen:

1. From the main menu, select the *Logic* menu item.
2. Choose *Processes*.
3. Choose *Header*.

## 4-4 Defining Application Logic



### Process Header Screen

#### Screen Description

This screen enables you to enter a brief description of the process for documentation purposes.

#### To enter the field values:

Field	Entry	Explanation
Process Name	product_proc	The name of the process within the application.
Description	Enter a suitable description.	Used for documentation purposes.

#### To complete and commit the screen:

1. Press the **Commit Data** function key to create the process header.

Once you've done this, go to the *Process Details* screen to create the process logic block.

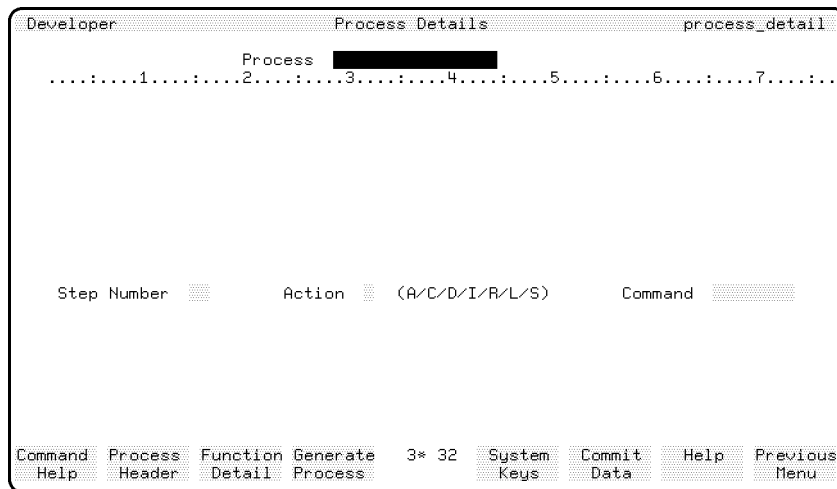
## Task 2 - Defining the Process Details

Now that you have created the process header, you can define the details of the process.

### Menu Path

To access the screen from the main menu:

1. Select the *Logic* option.
2. Choose *Processes*.
3. Choose *Details*.



**Process Details Screen**

### Screen Description

This screen allows you to create the command lines that make up the process logic block. The screen is divided into two distinct regions.

The upper region displays 10 of the 99 lines allowed in the logic block. This range is scrolled up and down to display the current line in the logic block as you work through it.

## 4-6 Defining Application Logic

The lower region allows you to enter commands. Three fields are always displayed in the lower region. These are the current step number, action, and command name. Below this a window is sometimes displayed, and this is where the remainder of the command line is entered or displayed.

### **Process Details Windows**

HP ALLBASE/4GL displays special windows for commands that have a more complex syntax. This special window steps you through the syntax of the particular command. HP ALLBASE/4GL displays the window after you enter a command name in the *Command* field.

The simpler commands use a free format window for you to enter the rest of the command. Some commands don't have options, so no window is displayed.

You will see examples of each of these methods as you enter the steps for this process.

### **About the Process**

Because each data manager is different, sometimes the logic that you write will vary according to the data manager you are using. In particular, commands that read or write to a data manager may vary.

This process, and some other processes and functions that you will create later, is slightly different for each data manager. Follow the instructions for the appropriate data manager. If you wish, take a look at the logic for other data managers to see how the logic varies for the needs of each data manager.

If you are developing the KSAM based application, read on below.

If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications.

If you are developing the HP TurboIMAGE/iX based application, turn to HP TurboIMAGE/iX Based Applications.

### **KSAM Based Applications**

For KSAM based applications, this is the process you are going to create.

- 1 **MODE** \*WRITE product
- 2 **SCREEN** product\_scrn
- 3 **FILE** \*WRITE product

## 4 ENTER 2

### What the Process Does

This process is quite simple.

- In the first line the `MODE` command specifies how the application can access data files. In this case, the application has write access to the data file called *product*.
- The `SCREEN` command in the second line displays the *product\_scrn* screen.
- The `FILE` command in the third line writes data from the file record buffer to the *product* file.
- The `ENTER 2` command in the last line causes the process to execute again from step number 2.
- The user can exit from the process by pressing the `Previous Menu` function key at any time.

```
Developer          Process Details          process_detail
                Process product_proc
.....1.....2.....3.....4.....5.....6.....7.....
1 MODE *WRITE product
2 SCREEN product_scrn
3 FILE *WRITE product
4 ENTER 2
5
6
7
8
9
10
Step Number 5      Action (A/C/D/I/R/L/S)      Command
Enter a step number or press the <Return> key for details of this step. (14108)
Command Process Function Generate 16* 18 System Commit Help Previous
Help Header Detail Process Keys Data Menu
```

**KSAM Process Details Screen**

## 4-8 Defining Application Logic



Enter the steps for this process as shown below.

**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Process Name</b>	product-proc	The default name in this field, if you have just defined the header for the process.

**Using Open Window Commands**

The first two commands use the open window method for you to enter the remainder of the step.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Step Number</b>	Accept the default.	The number of the step in the logic block that you are defining.  It is not necessary for step numbers to be contiguous. You can leave blank lines within a logic block.
<b>Action</b>	Accept the default.	An action code much like the one used to create the record layout details. It defaults to <i>A</i> indicating <i>add</i> for a new step, or <i>C</i> indicating <i>change</i> for an existing step.  The other valid actions available are <i>D</i> to <i>delete</i> a step, <i>I</i> to <i>insert</i> a step, <i>R</i> to <i>replace</i> a step, <i>L</i> to <i>list</i> the whole logic block, and <i>S</i> to <i>swap</i> the display pages if your logic block is longer than the ten lines displayed in the window.

Field	Entry	Explanation
Command	MODE	<p>When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command. HP ALLBASE/4GL automatically upshifts a lower-case entry.</p> <p>The MODE command sets the usage mode for a file while a process is active. By default, HP ALLBASE/4GL opens files for read access only. Refer to the <i>HP ALLBASE/4GL Developer Reference Manual</i> for more information about this command and any others you encounter in this manual.</p> <p>Pressing the <b>Command Help</b> function key on the process (or function) details screens displays several further sets of function keys that allow you to select the help screen you want.</p>
To Complete the Command	*WRITE product	<p>Specifies that you want to write to the <i>product</i> file. Note that you don't need to enter the word <i>write</i> in uppercase letters.</p>

#### 4-10 Defining Application Logic

**To complete this command:**

1. Press `Commit Data`.

This commits the step. HP ALLBASE/4GL clears the window and shows the new command line in the display region of the screen.

The cursor returns to the *Step Number* input field so you can proceed to enter the next command. The step number is automatically incremented as you build the logic block.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default value.	Notice that it is automatically incremented as you create each line of the logic block. If the current step is not displayed in the upper region of the screen, then the logic block is scrolled to ensure that the step is displayed.
Action	Accept the default action.	
Command	SCREEN	
To Complete the Command	Enter <code>product_scrn</code>	This command tells HP ALLBASE/4GL to display the <i>product_scrn</i> screen. After the process is executed, control returns to the process when the end user terminates the screen by pressing the <code>Commit Data</code> function key on this screen.

**To complete this command:**

1. Press `Commit Data`.

This displays the second command line in the display region of the screen. It also clears the command window and returns the cursor to the *Step Number* input field.

**To enter the next process:**

This process uses a command sensitive window to enter the remainder of the command line.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Step Number</b>	Accept the default.	
<b>Action</b>	Accept the default action.	
<b>Command</b>	Enter <b>FILE</b>	Displays a special window to prompt for further details. The window appears after you enter the command name <i>FILE</i> .
	<b>*WRITE</b>	Try entering an invalid file operation. HP ALLBASE/4GL displays a message informing you that the operation is invalid. The cursor remains on the field and you must correct the entry before you can move on.  This shows you how the command sensitive window helps you enter the correct command structure. Also note that you don't need to enter the asterisk before the command name. HP ALLBASE/4GL enters it for you.  Since the file used in this process has unique primary key values, it is possible to use the *WRITE option to both add or modify a record in the file.

**4-12 Defining Application Logic**

Field	Entry	Explanation
*NOLOCK	Accept the default.	Specifies whether this command imposes a lock on the file as it is accessed. The <i>N</i> option specifies that normal locking applies. (That is, the <i>NOLOCK</i> option is not used.)
File ID[.Record]	Enter <b>product</b>	The name of the file that is used for this operation. You can append a record specifier to the file name. This is only necessary if you want to access a file using a record layout other than the default record layout.

The remaining fields on the FILE command window are optional. You will use some of these fields later in this training course. For the moment, leave them blank.

**To complete this command:**

1. Press **Commit Data**.

When you press this key, HP ALLBASE/4GL clears the window and displays the command on the third line of the process logic block. Notice that the fields for which you made no entry are omitted from the command.

**To enter the fourth command:**

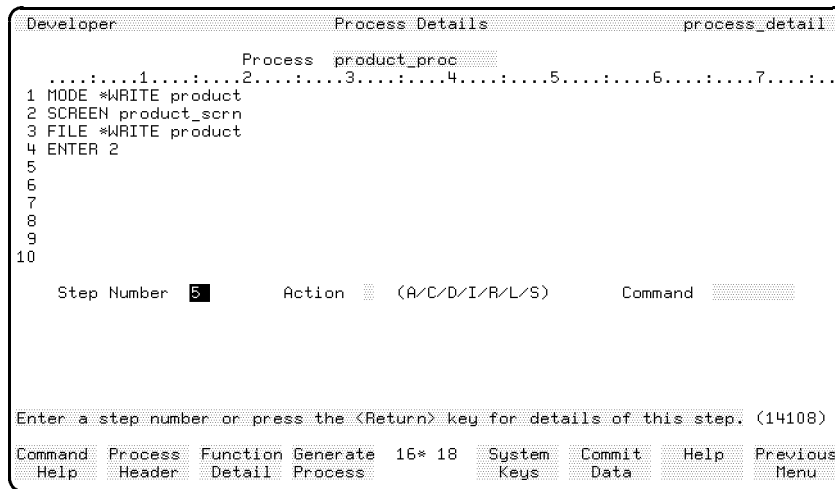
Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default action.	

Field	Entry	Explanation
Command	ENTER	When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.  The ENTER command passes control to another line of the process to execute it again.
To Complete the Command	2	This specifies that you want to pass control to line 2 of the process.

**To complete this command:**

1. Press `Commit Data`.

The new command line is displayed and the cursor returns to the *Step Number* field.



**Completed Process Details Screen**

The next step in creating a process is to generate it. This procedure is the same for all data managers. Turn to “Generating a Logic Block” to continue the training course and to generate the process.

**4-14 Defining Application Logic**

## Creating an SQL Process

For HP ALLBASE/SQL based applications, you are going to create the following process.

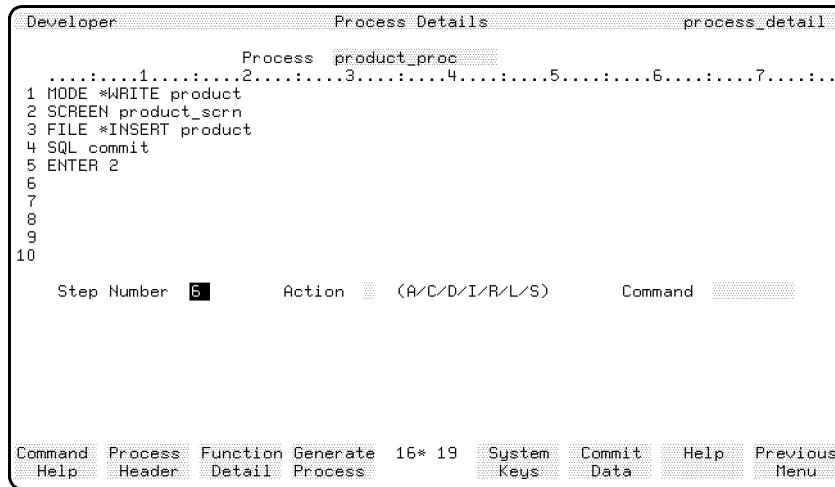
```
1 MODE *WRITE product
2 SCREEN product_scrn
3 FILE *INSERT product
4 SQL commit
5 ENTER 2
```

### What the Process Does

This process is quite simple.

- In the first line the **MODE** command specifies how the application can access data files. In this case the application has write access to the data file called *product*.
- The **SCREEN** command in the second line displays the *product\_scrn* screen.
- The **FILE** command in the third line inserts data from the file record buffer to the *product* file.
- The **SQL** command calls an SQL logic block to commit the work. You will create the SQL logic block in Lesson 8.
- The **ENTER** command in the last line causes the process to execute again from step number 2. The user can exit from the process by pressing the **Previous Menu** function key at any time.

The screen that follows shows how the finished process will look.



### Completed ALLBASE/SQL Process Details

#### To enter the field values:

You can now enter the data required to define the process. Each entry is detailed below.

Field	Entry	Explanation
Process Name	product_proc	This will be the default name in this field if you have just defined the header for the process.

#### Using Open Window Commands

The first two commands use the open window method for you to enter the remainder of the step.

## 4-16 Defining Application Logic



Field	Entry	Explanation
<b>Step Number</b>	Accept the default.	<p>The number of the step in the logic block that you are defining.</p> <p>It is not necessary for step numbers to be contiguous. You can leave blank lines within a logic block.</p>
<b>Action</b>	Accept the default.	<p>This is an action code much like the one you used to create the record layout details. It defaults to <i>A</i> indicating <i>add</i> for a new step, or <i>C</i> indicating <i>change</i> for an existing step.</p> <p>The other valid actions available are <i>D</i> to <i>delete</i> a step, <i>I</i> to <i>insert</i> a step, <i>R</i> to <i>replace</i> a step, <i>L</i> to <i>list</i> the whole logic block, and <i>S</i> to <i>swap</i> the display pages if your logic block is longer than the ten lines displayed in the window.</p>
<b>Command</b>	MODE	<p>When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.</p> <p>Notice that you don't have to enter upper-case characters. HP ALLBASE/4GL automatically upshifts a lower-case entry.</p> <p>The MODE command sets the usage mode for a file while a process is active. By default, HP ALLBASE/4GL opens files for read access only.</p> <p>You can get a brief description of a command by pressing the <b>Command Help</b> function key on the process (or function) details screens. This function key displays several further sets of function keys that allow you to select the help screen you want.</p>

Field	Entry	Explanation
To Complete the Command	*WRITE product	This specifies that you want to write to the <i>product</i> file. Note that you don't need to enter the word <i>write</i> in uppercase letters.

**To complete this command:**

1. Press `Commit Data` .

This commits the step. HP ALLBASE/4GL clears the window and shows the new command line in the display region of the screen. The cursor returns to the *Step Number* input field so you can proceed to enter the next command. The step number is automatically incremented as you build the logic block.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	Notice that it is automatically incremented as you create each line of the logic block. If the current step is not displayed in the upper region of the screen, then the logic block is scrolled to ensure that the step is displayed.
Action Command	Accept the default. SCREEN	
To Complete the Command	product_scrn	This command tells HP ALLBASE/4GL to display the <i>product_scrn</i> screen.  When the process is executed, control returns to the process when the end user terminates the screen by pressing the <code>Commit Data</code> function key on this screen.

**To complete this command:**

1. Press `Commit Data` .

**4-18 Defining Application Logic**

This displays the second command line in the display region of the screen. It also clears the command window and returns the cursor to the *Step Number* input field.

### A Special Window Command

The next step uses a command sensitive window to enter the remainder of the command line.

Field	Entry	Explanation
<b>Step Number</b>	Accept the default.	
<b>Action</b>	Accept the default.	
<b>Command</b>	<b>FILE</b>	The file command displays a special window to prompt for further details. The window appears after you enter the command name <i>FILE</i> .
	<b>*INSERT</b>	If you enter an invalid file operation, HP ALLBASE/4GL displays a message informing you that the operation is invalid. The cursor remains on the field and you must correct the entry before you can move on.  This shows you how the command sensitive window helps you enter the correct command structure. Also note that you don't need to enter the asterisk before the command name. HP ALLBASE/4GL enters it for you.
<b>*NOLOCK</b>	Accept the default.	Specifies whether this command imposes a lock on the file as it is accessed. The <i>N</i> option specifies that normal locking applies. (That is, the <i>NOLOCK</i> option is not used.)

Field	Entry	Explanation
File ID[.Record]	product	The name of the file that is used for this operation. You can append a record specifier to the file name. This is only necessary if you want to access a file using a record layout other than the default record layout.

### The Remaining Fields

The remaining fields on the FILE command window are optional. You will use some of these fields later in this training course. For the moment, leave them blank.

#### To complete this command:

1. Press `Commit Data`.

When you press this key, HP ALLBASE/4GL clears the window and displays the command on the third line of the process logic block. Notice that the fields for which you made no entry are omitted from the command.

#### To enter the next command:

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	SQL	This displays an open window.

## 4-20 Defining Application Logic

Field	Entry	Explanation
To Complete the Command	commit	<p>This command calls an SQL logic block called <i>commit</i>. You will create this SQL logic block in Lesson 8.</p> <p>HP ALLBASE/SQL requires you to issue an explicit COMMIT WORK to complete a transaction and make it permanent in the database. If you terminate a transaction or attempt to start a new transaction without issuing a commit work command, HP ALLBASE/SQL reverses any currently open transaction. The <i>commit</i> SQL logic block will issue the COMMIT WORK command.</p>

**To complete this command:**

1. Press `Commit Data` .

This displays the fourth command line in the display region of the screen. It also clears the command window and returns the cursor to the *Step Number* input field.

**To enter the final command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	ENTER	<p>When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.</p> <p>The ENTER command passes control to another line of the process to execute it again.</p>
To Complete the Command	2	Specifies that you want to pass control to line 2 of the process.

**To complete this command:**

1. Press `Commit Data`.

The new command line is displayed and the cursor returns to the *Step Number* field.

**Generating the Process**

The next step in creating a process is to generate it. This procedure is the same for all data managers. Turn to the section “Generating a Logic Block” to continue the training course and to generate the process.

**Notes on Entries****SQL Command Equivalent**

The FILE \*INSERT command inserts the current contents of the *product* file buffer into the *product* table as a new row.

This command is equivalent to the SQL command:

```
INSERT INTO sqlgrp.product
VALUES (:F-product_no.product,
       :F-description.product,
       :F-supplier_no.product,
       :F-lead_time.product);
```

In this application, the user cannot attempt to insert a row into the *product* table if a row with the same value in the *product\_no* field already exists. If the application logic allows the user to attempt an insert operation that causes an index uniqueness violation, HP ALLBASE/SQL generates an error message. If the FILE command has an optional error command, control would pass to the error command if such an error occurred.

**Creating an HP TurboIMAGE/iX Process**

For HP TurboIMAGE/iX based applications, this is the process you are going to create.

```
1 DM IMAGE *MODE *MODLOCK :D-traindb
2 MODE *WRITE product
```

**4-22 Defining Application Logic**

```
3 SCREEN product_scrn
4 FILE *WRITE product
5 DM IMAGE *UNLOCK :D-traindb
6 ENTER 3
```

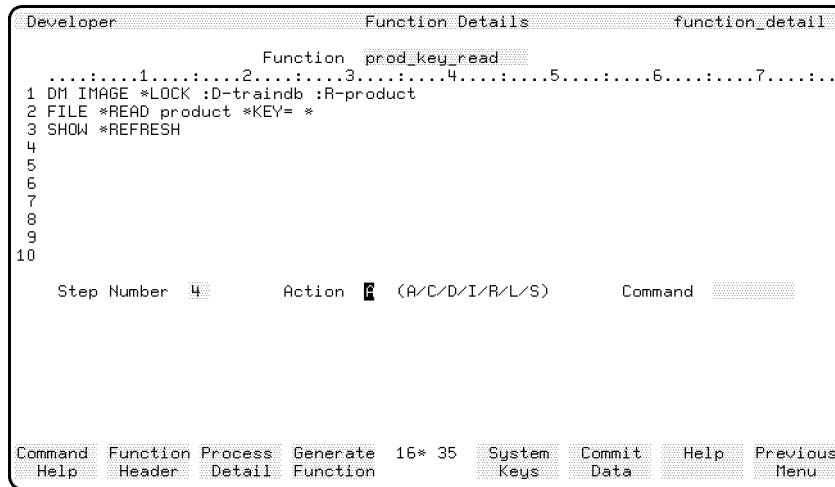
### What the Process Does

This process is quite simple.

- In the first line the DM IMAGE
- MODE command specifies how the application can access the *traindb* database. In this case the application has read access, and write access if the part of the database that you wish to write to is locked by the application.
- In the second line, the MODE \*WRITE command also specifies how the application can access the data, but this is specific to the *product* data set only. This command must be used in conjunction with the DM IMAGE \*MODE command, and must not offer greater capability than the DM IMAGE
- MODE command.
- The SCREEN command in the third line displays the *product\_scrn* screen.
- The FILE command in the fourth line writes data from the file record buffer to the *product* file.
- When the screen field *product\_no* is entered, a function *product\_key\_read* is called that locks and reads the data set. The DM IMAGE \*UNLOCK command releases all locks held by this process on the *traindb* database.

In this case, the lock on the record placed by the *product\_key\_read* function is the only lock released. The lock on the data set is held from the time the user enters a product number until after the user presses the **Commit Data** function key.

- The ENTER 3 command in the last line causes the process to execute again from step number 3. The user can exit from the process by pressing the **Previous Menu** function key at any time. The process should be entered as shown below.



### Completed HP TurboIMAGE/iX Process Details Screen

To enter the field values:

Field	Entry	Explanation
Process Name	product_proc	This will be the default name in this field if you have just defined the header for the process.

### Open Window Commands

The first three commands use the open window method for you to enter the remainder of the step.

## 4-24 Defining Application Logic



<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Step Number</b>	Accept the default.	<p>The number of the step in the logic block that you are defining.</p> <p>It is not necessary for step numbers to be contiguous. You can leave blank lines within a logic block.</p>
<b>Action</b>	Accept the default.	<p>This is an action code much like the one you used to create the record layout details. It defaults to <i>A</i> indicating <i>add</i> for a new step, or <i>C</i> indicating <i>change</i> for an existing step.</p> <p>The other valid actions available are <i>D</i> to <i>delete</i> a step, <i>I</i> to <i>insert</i> a step, <i>R</i> to <i>replace</i> a step, <i>L</i> to <i>list</i> the whole logic block, and <i>S</i> to <i>swap</i> the display pages if your logic block is longer than the ten lines displayed in the window.</p>

Field	Entry	Explanation
Command	DM	<p>When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.</p> <p>Notice that you don't have to enter upper-case characters. HP ALLBASE/4GL automatically upshifts a lower-case entry.</p> <p>The DM command is used as a prefix to logic commands specific to a certain data manager. Currently, HP ALLBASE/4GL only offers data manager specific commands for the HP TurboIMAGE/iX data manager.</p> <p>You can obtain a brief description of a command by pressing the <b>Command Help</b> function key on the process (or function) details screens. This function key displays several further sets of function keys that allow you to select the help screen you want.</p>

#### 4-26 Defining Application Logic

Field	Entry	Explanation
To Complete the Command	IMAGE *MODE *MODLOCK :D-traindb	The DM IMAGE *MODE command specifies how the application can access one or more databases.  In this case, you are setting the access mode for the <i>traindb</i> database. Each access mode represents an HP TurboIMAGE/iX access mode. The access mode *MODLOCK is equivalent to HP TurboIMAGE/iX access mode 1. This mode gives the application read access, and write access if the part of the database to be altered is locked before being altered. It is not necessary to use a lock to read the database.

**To complete this command:**

1. Press **Commit Data** .

This commits the step. HP ALLBASE/4GL clears the window and shows the new command line in the display region of the screen.

The cursor returns to the *Step Number* input field so you can proceed to enter the next command. The step number is automatically incremented as you build the logic block.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	Notice that it is automatically incremented as you create each line of the logic block. If the current step is not displayed in the upper region of the screen, then the logic block is scrolled to ensure that the step is displayed.

Field	Entry	Explanation
Action	Accept the default.	
Command	MODE	
To Complete the Command	*WRITE product	This command tells HP ALLBASE/4GL that it may execute any commands that require write access to the product data set.

**To complete this command:**

1. Press `Commit Data`.

This displays the second command line in the display region of the screen. It also clears the command window and returns the cursor to the *Step Number* input field.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	SCREEN	
To Complete the Command	product_scrn	This command tells HP ALLBASE/4GL to display the <i>product_scrn</i> screen. When the process is executed, control returns to the process when the end user terminates the screen by pressing the <code>Commit Data</code> function key on this screen.

**To complete this command:**

1. Press `Commit Data`.

This displays the third command line in the display region of the screen. It also clears the command window and returns the cursor to the *Step Number* input field.

**4-28 Defining Application Logic**

### Using a Special Window Command

The next step uses a command sensitive window to enter the remainder of the command line.

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	FILE	The file command displays a special window to prompt for further details. The window appears after you enter the command name <i>FILE</i> .
	*WRITE	If you enter an invalid file operation, HP ALLBASE/4GL displays a message informing you that the operation is invalid. The cursor remains on the field and you must correct the entry before you can move on. This shows you how the command sensitive window helps you enter the correct command structure. Also note that you don't need to enter the asterisk before the command name. HP ALLBASE/4GL enters it for you.  Since the file used in this process has unique primary key values, it is possible to use the *WRITE option to both add or modify a record in the file.

Field	Entry	Explanation
<b>*NOLOCK</b>	Accept the default.	Specifies whether this command imposes a lock on the file as it is accessed. The <i>N</i> option specifies that normal locking applies. (That is, the <i>NOLOCK</i> option is not used.)
<b>File ID[.Record].</b>	<b>product</b>	The name of the file that is used for this operation. You can append a record specifier to the file name. This is only necessary if you want to access a file using a record layout other than the default record layout.

### Remaining Fields

The remaining fields on the FILE command window are optional. You will use some of these fields later in this training course. For the moment, leave them blank.

#### To complete this command:

1. Press **Commit Data** .

When you press this key, HP ALLBASE/4GL clears the window and displays the command on the third line of the process logic block. Notice that the fields for which you made no entry are omitted from the command.

#### To enter the second DM command:

Field	Entry	Explanation
<b>Step Number</b>	Accept the default.	
<b>Action</b>	Accept the default.	
<b>Command</b>	<b>DM</b>	When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.

### 4-30 Defining Application Logic

Field	Entry	Explanation
To Complete the Command	IMAGE *UNLOCK :D-traindb	This command unlocks all locks held by the process on the <i>traindb</i> database or on any part of the <i>traindb</i> database.  On this occasion there is only one lock; this lock is created in the <i>product_key_read</i> function that you will create soon. The function is called from the <i>product_scrn</i> screen and is executed while the process is active, so it is classified as a lock held by the process.

**To complete this command:**

1. Press **Commit Data** .

The new command line is displayed and the cursor returns to the *Step Number* field.

**To enter the last command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	ENTER	When you enter this command name, HP ALLBASE/4GL presents a free form data entry area at the base of the screen so you can enter the remainder of the command.  The ENTER command passes control to another line of the process to execute it again.
To Complete the Command	3	This specifies that you want to pass control to line 3 of the process.

**To complete this command:**

1. Press **Commit Data** .

The new command line is displayed and the cursor returns to the *Step Number* field.

### **Generating the Process**

The next step in creating a process is to generate it. This procedure is the same for all data managers.

---

## **Task 3 - Generating the Process**

Like data screens and record layouts, processes and functions also require generation. Generating the process checks for the existence of the data names, records, and files you have named, and resolves pointers to these items.

### **To generate the process:**

1. Press the **Generate Process** function key to generate the *product\_proc* process.

If you entered the process correctly, you won't receive any error messages.

If you have made an error, you will receive a message listing the step number and character position value that was found to be in error, and a brief description of the error. The error message is displayed for about 20 seconds. HP ALLBASE/4GL then returns you to the details screen.

2. You can press the **Stop** key to freeze the display.

The *step number* column refers to the step number in the logic block, the *character position value* column refers to the ruler line at the top of the display region of the logic block screen. By using the ruler line at the top of the screen, you can determine the horizontal location of the incorrect value.

3. Once you have determined the cause of the error, you can change the step by entering its step number and accepting the *C* action that is displayed by default.

This displays the command line at that step in the appropriate window, and you can correct the error. (If the error is in the Command field, press **Shift** + **Tab** to move to this field from the window.) Once you have corrected the

## **4-32 Defining Application Logic**



entry, press the **Commit Data** function key, and generate the logic block again.

When the process has been successfully generated, you can continue on to develop a function for this application.

---

## Summary

In this lesson you have created a process.

A process is similar to a program in a conventional programming system. It terminates any current activity in the application, and initializes the environment to a known state.

The process you have created displays the *product\_scrn* data screen and updates or inserts the *product* data file when the user terminates the screen.

You will see this scenario of a process driving a data screen, which in turn calls a function, repeated many times as you develop applications.

There are three steps involved in developing a process. They are:

1. Defining the process header.
2. Defining the process details.
3. Generating the process.

---

## Lesson 7 - Developing a Function

---

### Objectives

When you have completed this lesson you will have learned how to develop a function. You will have learned how to perform these tasks:

1. Defining a function header.
2. Defining the function details.
3. Generating the function.

---

### Functions

Functions are similar to subroutines in a conventional programming language. When a function ends, control returns to the item that called the function. Functions use the same constructs, and generally the same commands, as processes.

The function you are about to create is executed by the screen logic as an *after* function for a field on the screen. This means that it is executed **after** the user enters data to the screen field.

Functions can also be called from a function key or through a VISIT command in any logic block, as well as a number of other methods. It's important to remember that a function does not alter the application environment when it's called.

Developing a function is similar to developing a process. You must define the header first, and then the details. A function must be generated before the application can use it.

Every logic command except the MODE command can be used in a function. The MODE command is only permitted in a process.

#### 4-34 Defining Application Logic

---

## Task 1 - Defining the Function Header

The function you are about to create is executed on completion of data entry for the *product\_no* field on the *product\_scrn* data screen. The function reads the product file using the value just entered as the key for the file search, and then displays the data from the file if a matching record is found. For HP ALLBASE/SQL based applications, the function initially calls an SQL logic block to declare and open a cursor on a matching record, if it is found.

### Menu Path

#### To access the function header screen:

1. From the main menu, select the *Logic* option.
2. Select *Functions*.
3. Choose *Header*.

### Screen Description

The Function Header Screen lets you enter a brief description of the function for documentation purposes.

The screenshot shows a terminal window titled "Function Header" with the following content:

```
Developer      Function Header      function_header

Function Name   product_key_read

Description
product_key_read
AUTHOR: developr
This function is called after data is entered in the
product number field on the product_scrn data screen.
Last Modification:   Date           Time

Command  Function Process  SQL Blk  5* 33  System  Commit  Help  Previous
Help    Detail  Menu     Menu   Keys   Data   Menu
```

**Function Header Screen**

**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Function Name</b>	product_key_read	The name of the function within the application.
<b>Description</b>	Enter a suitable description.	

**To complete and commit this screen:**

1. Press the **Commit Data** function key to create the function header.

Now go to the function detail screen to create the logic block commands.

---

## **Task 2 - Defining the Function Details**

Having created the function header, you can now define the function details.

### **Menu Path**

**To access this screen from the main menu:**

1. Select the *Logic* option.
2. Select the *Functions* option.
3. Choose *Details*.

### **Screen Description**

This screen is identical in appearance and operation to the process details screen. The only difference is that you are creating a function, not a process. It uses the same step numbers, actions, and commands as the process details screen and it behaves in a similar manner.

The function you will create differs according to the data manager you are using.

Read on if the application you are developing is KSAM based.

## **4-36 Defining Application Logic**

Turn to HP ALLBASE/SQL Based Application if you are developing the HP ALLBASE/SQL based application.

Turn to HP TurboIMAGE/iX Based Application if you are developing the HP TurboIMAGE/iX based application.

## Developing a KSAM Function

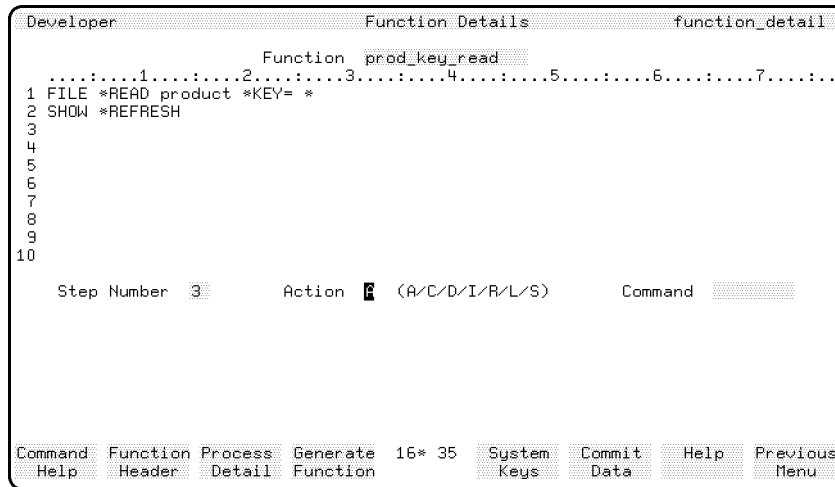
This is the function that you will create:

```
1 FILE *READ product *KEY=*  
2 SHOW *REFRESH
```

When the user enters data into the *product\_no* field on the screen and presses **Return**,

- The FILE command searches the *product* data file for a record that has a key that matches the data entered by the user.
- If a matching record is found, the SHOW command displays the data contained in the file record on the screen.
- If no match is found, the SHOW command displays the entered data again, and the user can continue to complete the screen.

The screen image shows you what the screen will look like when you have entered the function.



### Completed Function Details Screen

To enter the field values:

Field	Entry	Explanation
Function Name	product_key_read	This will be the default value if you have just created the header for this function.
Step Number	Accept the default.	
Action	Accept the default.	
Command	FILE	You will see the FILE window appear. The window is exactly the same as the one you used when you were defining the process.
File Operation	*READ	This time you want to read a record from the file. You can enter this command as READ without the asterisk.

#### 4-38 Defining Application Logic

Field	Entry	Explanation
*NOLOCK	Accept the default.	
File ID[.Record]	product	This is a reference to the product file using the default record layout.
*INDEX=	Leave blank.	This field allows you to specify which index is used to access the file. Leaving it blank means that the file is accessed using the index specified by the current value in the communication area field *INDEXNO.  HP ALLBASE/4GL initializes the value in *INDEXNO to 1 when an application starts, assuming that all file access is to be via the primary key.
*KEY=	*	Specifies the value to be moved to the key field of the file buffer before the file read is performed.  The * refers to the internal buffer of the current screen field. As this function is called from the <i>product_no</i> field, the * refers to the contents of the <i>product_no</i> field on the data screen.  If you don't specify a key, the file read takes place using the current contents of the key field in the file buffer.

**To complete this command:**

1. Press **Commit Data** .

When you press this key, HP ALLBASE/4GL clears the window and displays the entered command at the first line of the function. You will see that the \*KEY= \* clause is included in the command line.

**To enter the next command:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Step Number</b>	Accept the default.	
<b>Action</b>	Accept the default.	
<b>Command</b>	SHOW	This command redisplay the fields on the screen.
<b>To Complete the Command</b>	*REFRESH	Notice that you can enter either lowercase or uppercase characters for the word REFRESH. The *REFRESH clause specifies that data is moved from the data movement fields to the internal screen buffers before the fields are redisplayed.  All the fields on the screen have been created with primary movement fields from the product file. This means that the current contents of the file buffer are displayed by this command.

**To complete this command:**

1. Press `Commit Data`.

HP ALLBASE/4GL displays the new command line and clears the window. This function is now complete.

To generate the function, refer to the “Generating A Function” section following the instructions for SQL and TurboIMAGE/iX.

**4-40 Defining Application Logic**



## Developing an SQL Function

For an HP ALLBASE/SQL application, this is the function that you will create:

```
1 SQL find_prod
2 FILE *NEXT product
3 SHOW *REFRESH 2 4
```

When the user enters data into the *product\_no* field on the screen and presses ,

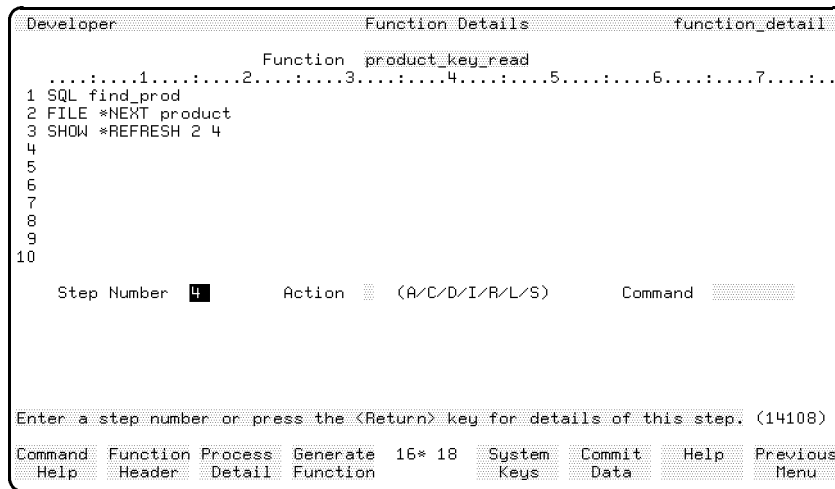
- the first step executes the SQL commands in the *find\_prod* SQL logic block. This SQL block contains a SELECT command to retrieve the record matching the product number value entered by the user. You will create this SQL logic block in the next lesson.
- The FILE \*NEXT command following the SQL command is equivalent to an SQL FETCH command. This command positions the cursor on the first row of the active set, and retrieves the record into the file buffer for the *product* table.
- The remaining commands in the logic block then use this data as required. If the active set for the cursor is empty, the FILE \*NEXT command encounters an end-of-file condition.

This example demonstrates the standard technique for retrieving SQL data from an application. To retrieve a record you must perform the following steps:

1. Use a SELECT command in an SQL logic block to declare and open a cursor.
2. Use a FILE \*NEXT command in a logic block to position the cursor on the first record, and retrieve the record into the file buffer.

Both these steps must be in the same process, although they do not need to be in the same logic block. For example, you can use the SELECT command in an SQL logic block called from one function, and the FILE \*NEXT command in another function, provided both functions are called from the same process. You cannot use a SELECT command in one process, and the FILE \*NEXT command in another process because HP ALLBASE/4GL automatically closes all open HP ALLBASE/SQL cursors at the end of a process.

The screen image shows you what the screen will look like when you have entered the function.



**Completed SQL Function Details Screen**

To enter the field values:

Field	Entry	Explanation
Function Name	product_key_read	This will be the default value if you have just created the header for this function.
Step Number	Accept the default.	
Action	Accept the default action.	
Command	SQL	This displays an open window.
To Complete the Command	find_prod	The name of the SQL logic block that you will define in the next lesson. The <i>find_prod</i> SQL logic block will declare and open a cursor in the <i>product</i> table.

To complete this command:

1. Press **Commit Data** .

#### 4-42 Defining Application Logic

When you press this key, HP ALLBASE/4GL clears the window and displays the entered command at the first line of the function.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default of 2..	
Action	Accept the default of A.	
Command	FILE	Again you will see the FILE window appear. The window is exactly the same as the one you used when you were defining the process.
File Operation	*NEXT	After placing the SQL cursor in the correct position, you want to retrieve the first record found by the SQL SELECT command. You can enter this command as NEXT without the asterisk.
*NOLOCK	Accept the default.	
File ID[.Record]	product	This is a reference to the product table.

**To complete this command:**

1. Press `Commit Data`.

When you press this key, HP ALLBASE/4GL clears the window and displays the entered command at the second line of the function.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	SHOW	

Field	Entry	Explanation
To Complete the Command	*REFRESH 2 4	<p>Notice that you can enter either lowercase or uppercase characters for the word REFRESH. This command redisplay the fields on the screen.</p> <p>The *REFRESH clause specifies that data is moved from the data movement fields to the internal screen buffers before the fields are redisplayed. All the fields on the screen have been created with primary movement fields from the product file. This means that the current contents of the file buffer are displayed by this command. The 2 4 portion of the SHOW command tells HP ALLBASE/4GL only to refresh fields 2 to 4 of the screen.</p>

**To complete this command:**

1. Press `Commit Data`.

HP ALLBASE/4GL displays the new command line and clears the window. This function is now complete.

To generate the function, refer to “Task 3 - Generating the Function” following the instructions for TurboIMAGE/iX.

**4-44 Defining Application Logic**

## Developing an HP TurboIMAGE/iX Function

This is the function that you will create:

```
1 DM IMAGE *LOCK :D-traindb :R-product
2 FILE *READ product *KEY=*
3 SHOW *REFRESH
```

When the user enters data into the *product\_no* field on the screen and presses **Return**,

- The DM IMAGE \*LOCK command places a *logical* HP TurboIMAGE/iX lock on the *product* data set.
- Then the FILE command searches the *product* data set for a record that has a key that matches the data entered by the user.
- If a matching record is found, the SHOW command displays the data contained in the file record on the screen.

If no match is found, the SHOW command displays the entered data again, and the user can continue to complete the screen.

The screen image shows you what the screen will look like when you have entered the function.

```
Developer                               Function Details                       function_detail
                                     Function product_key_read
.....1.....2.....3.....4.....5.....6.....7.....
1 DM IMAGE *LOCK :D-traindb :R-product
2 FILE *READ product *KEY= *
3 SHOW *REFRESH
4
5
6
7
8
9
10

Step Number  4      Action  (A/C/D/I/R/L/S)  Command

Enter a step number or press the <Return> key for details of this step. (14108)
Command  Function  Process  Generate  15* 18  System  Commit  Help  Previous
Help    Header  Detail  Function  Keys    Data    Menu
```

**Completed HP TurboIMAGE/iX Function Detail Screen**

**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Function Name</b>	<code>product_key_read</code>	This will be the default value if you have just created the header for this function.
<b>Step Number</b>	Accept the default.	
<b>Action</b>	Accept the default.	
<b>Command</b>	<code>DM</code>	As with the DM commands that you defined in the <i>product_proc</i> process, an open window is displayed.
<b>To Complete the Command</b>	<code>IMAGE *LOCK</code> <code>:D-traindb</code> <code>:R-product</code>	<p>This command places a logical lock on the <i>product</i> data set in the <i>traindb</i> database.</p> <p>Each HP ALLBASE/4GL item reference is prefixed by a colon (:).</p> <p>The logical lock is a normal HP TurboIMAGE/iX lock, where the lock descriptors are stored in a table. All new lock requests are compared with the descriptors currently in the table. If the descriptor already exists in the table, the new lock is denied; otherwise it is granted.</p> <p>The lock does not check to ensure whether any records with the product number specified exist.</p> <p>This lock is maintained until the DM <code>IMAGE *UNLOCK</code> command is executed in the <i>product_proc</i> process.</p>

**To complete this command:**

1. Press `Commit Data`.

When you press this key, HP ALLBASE/4GL clears the window and displays the entered command at the first line of the function.

**4-46 Defining Application Logic**

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	Again you will see the FILE window appear. The window is exactly the same as the one you used when you were defining the process.
File Operation	*READ	This time you want to read a record from the file. You can enter this command as READ without the asterisk.
*NOLOCK	Accept the default.	
File ID[.Record]	product	Again, this is a reference to the default record layout on the product file.
*INDEX=	Leave blank.	This field allows you to specify which index is used to access the file. Leaving it blank means that the file is accessed using the index specified by the current value in the communication area field *INDEXNO.  HP ALLBASE/4GL initializes the value in *INDEXNO to 1 when an application starts, assuming that all file access is to be via the primary key.
*KEY=	*	This clause specifies the value to be moved to the key field of the file buffer before the file read is performed.  The * refers to the contents of the current field ( <i>product_no</i> ) on the data screen.  If you don't specify a key, the file read takes place using the current contents of the key field in the file buffer.

**To complete this command:**

1. Press `Commit Data` .

When you press this key, HP ALLBASE/4GL clears the window and displays the entered command at the second line of the function. You will see that the \*KEY= \* clause is included in the command line.

**To enter the next command:**

Field	Entry	Explanation
Step Number	Accept the default.	
Action	Accept the default.	
Command	SHOW	
To Complete the Command	*REFRESH	<p>Notice that you can enter either lowercase or uppercase characters for the word REFRESH. This command redisplay the fields on the screen. The *REFRESH clause specifies that data is moved from the data movement fields to the internal screen buffers before the fields are redisplayed.</p> <p>All the fields on the screen have been created with primary movement fields from the product file. This means that the current contents of the file buffer are displayed by this command.</p>

**To complete this command:**

1. Press `Commit Data` .

HP ALLBASE/4GL displays the new command line and clears the window. This function is now complete.

Now you can generate the function. This procedure is the same for all data managers.

**4-48 Defining Application Logic**



---

## Task 3 - Generating the Function

A function must be generated before it can be used in the application. The procedure for generating a function is identical to that used to generate a process.

### To generate a function:

1. Press the **Generate Function** function key.

HP ALLBASE/4GL displays a message to indicate that the function is being generated. The function will either generate correctly, or a listing of any errors encountered will be displayed.

2. If any errors are detected, you can correct them in the same way described for the process logic block error correction. Refer to Lesson 6.

---

## Summary

In this lesson you created a function.

The function is executed as an after function for the *product\_no* screen field, and performs a file lookup after the user enters data in the field.

There are three steps involved in developing a function. They are:

1. Define the function header.
2. Define the function details.
3. Generate the function.

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, you have completed all the logic you need for now. Turn to the Self Test Questions at the end of this chapter.

In Chapter 5, you can start testing the application.

If you are developing the HP ALLBASE/SQL based application, complete the next lesson to create the SQL logic blocks called by the *product\_proc* process and the *product\_key\_read* function.

---

## Lesson 8 - Creating an SQL Logic Block

### Objectives

When you have completed this lesson you will have learned how to develop two SQL logic blocks. If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, skip this lesson, and turn to the Self Test Questions.

To develop an SQL logic block you will perform these tasks:

- Define an SQL logic block header.
- Define the SQL logic details.
- Generate the SQL logic block.

---

### SQL Logic Blocks

The next few pages describe the procedures for creating SQL logic blocks, and outline some of the rules covering the permissible SQL commands for SQL logic blocks.

The *product\_key\_read* function uses the *find\_prod* SQL logic block, and the *product\_proc* process uses the *commit* SQL logic block. Now that you have successfully generated the process and the function, you will create the SQL logic blocks called by them.

#### 4-50 Defining Application Logic

---

## Defining an SQL Logic Block

Defining an SQL logic block is similar to defining a function. You must define an SQL logic block header, then define the details for the logic block, and finally, generate the logic block to convert it to an executable form.

The screens you use to define SQL logic blocks are accessible from the logic menu.

To create the *find\_prod* SQL logic block, you must first create the logic block header. To do this, display the SQL logic block header screen.

This screen is similar in appearance and operation to the function and process header screens.

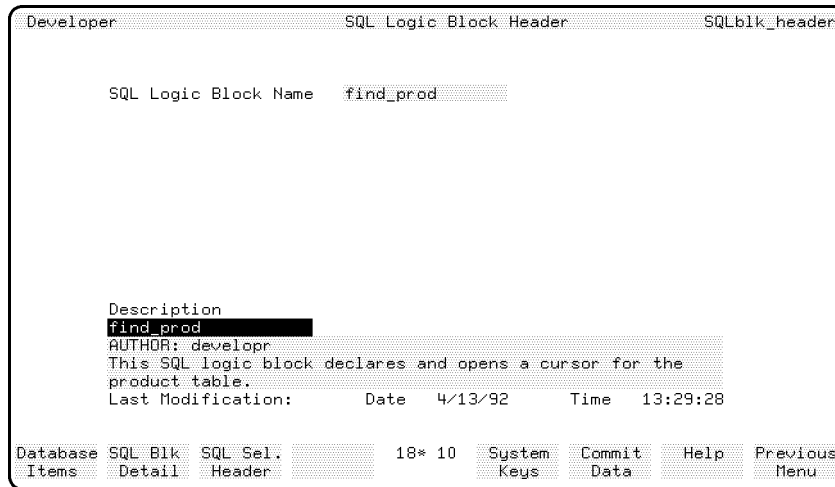
### Menu Path

#### To access this screen:

1. From the main menu, select the *Logic* option.
2. Select *SQL Logic Blocks*.
3. Choose *Header*.

### Screen Description

This screen enables you to enter a brief description of the SQL logic block for documentation purposes.



### SQL Logic Block Header Screen

To enter the field values:

Field	Entry	Explanation
SQL Logic Block Name	find_prod	This is the name of the SQL logic block within the applications.
Description	Enter a suitable description.	

To complete this command:

1. Press the **Commit Data** function key to create the SQL logic block header.

Now go to the *SQL Logic Block Details* screen to create the logic block commands.

## 4-52 Defining Application Logic

---

## Defining the SQL Logic Block Details

You can now create the details for the SQL logic block.

### Menu Path

**To access this screen from the main menu:**

1. Select the *Logic* option.
2. Select *SQL Logic Blocks*.
3. Choose *Details*.

### Screen Description

This screen contains a free format text entry area that allows you to enter the SQL commands that make up the SQL logic block.



**SQL Logic Block Details Screen**

### Entering the Details

You can use the normal terminal keyboard keys to enter and edit the commands in this text entry area.

The *find\_prod* SQL logic block contains a single SQL SELECT command as follows:

```
SELECT :product FROM sqlgrp.product
WHERE product_no = :S-product_no.product_scrn
FOR UPDATE OF description, supplier_no, lead_time;
```

**To enter the command:**

1. Enter the command in the text area.
2. Make sure that you include the colon (:) before the term *product* in the first line, and the term *S-product\_no.product\_scrn* in the second line.
3. Terminate the command with a semicolon (;) at the end of the last line of the command.

It doesn't matter which lines you enter the commands on, or whether you indent lines, as long as the commands are in the correct sequence.

4. When you have entered the command, press the **Commit Data** function key.

**The SELECT Command**

HP ALLBASE/4GL modifies the syntax of the SQL SELECT command slightly. In SQL logic blocks, you must enter the command in the following format.

```
SELECT { :table_name } FROM clauses [other clauses ]
      :select_list
```

```
[FOR UPDATE OF column_name [,column_name ...]];
```

In this expression, *table\_name* is an HP ALLBASE/SQL table name that has been defined on the file/SQL table definition screen in the dictionary, and *select\_list* is a select list name that has been defined in the dictionary. You cannot use the SELECT \* FROM ... form of the SELECT command in an SQL logic block, and you cannot use the INTO clause in a SELECT command.

At generate time, HP ALLBASE/4GL converts this command to a DECLARE CURSOR command in the format:

**4-54 Defining Application Logic**

```
DECLARE SQL_block_name CURSOR FOR select_command
FOR UPDATE OF .....
```

where *SQL\_block\_name* is the name of the SQL logic block containing the SELECT command.

## Host Variables

You can include references to the following data items in SQL logic blocks.

- File record fields and work area fields.
- Variables and calculated items.
- Numeric and alphanumeric constants.
- Screen fields.

This is referred to as using host variables. Note that you cannot use host variables to replace the name of an SQL table.

References to HP ALLBASE/4GL data items in SQL logic blocks must be prefixed with a colon (:). The reference must be in the format:

*:data\_ref*

where *data\_ref* is the full reference to the item.

For example, the term *:S-product\_no.product\_scrn* in the SQL logic block you have just created is a reference to the *product\_no* field on the *product\_scrn* screen. (Note that references to screen fields in SQL logic blocks must be fully qualified references by name.) Refer to the *HP ALLBASE/4GL Developer Reference Manual* for more information about the referencing rules.

## Generating an SQL Logic Block

You can now generate the *find\_prod* SQL logic block.

1. Start the generation process by pressing the **Generate SQL** function key.

HP ALLBASE/4GL calls the generate program, and then connects to the database environment for the application.

During SQL logic block generation, HP ALLBASE/4GL validates and resolves references to data items. The generation process also converts the SQL

commands in the SQL block to an executable form and stores them in the database environment as stored sections in a database module.

In its generated form, the *prod\_find* command is equivalent to the following SQL DECLARE CURSOR command:

```
DECLARE find_prod CURSOR FOR
  SELECT * FROM sqlgrp.product
  WHERE product_no = :S-product_no.product_scrn;
```

At run-time HP ALLBASE/4GL executes this command as a DECLARE CURSOR command, followed by an OPEN CURSOR command.

The effect of this command is to declare and open a cursor on the *product* table. The active set for the cursor is the set of rows in the table that match the search condition specified by the WHERE clause in the SELECT command. In this particular case, the active set consists of a maximum of one record because the *product\_no* field is defined as a unique index to the table. If no record matches the search condition, the active set for the cursor is empty.

### **Generation Errors**

If HP ALLBASE/4GL detects any errors in an SQL logic block during generation, it displays a generate error screen.

The generate error screen may specify a step number and a character position for the error. For SQL logic blocks, this position refers to the cursor position at or near the cause of the error. (Most HP terminals display the current cursor position in terms of the screen line number and column number between the two blocks of function key labels.)

If you receive an error message when you generate this SQL block, check that you have entered it correctly. Make sure that you have terminated the command with a semicolon, and that you have included the colons where they are required.

## **4-56 Defining Application Logic**



## SQL Logic Block Limitations

The *find\_prod* SQL logic block introduces a number of features and limitations that apply to all SQL logic blocks.

The next few paragraphs describe these items.

- Some SQL commands are not permitted in SQL logic blocks. The *HP ALLBASE/4GL Developer Reference Manual* contains a full list of the commands you cannot use.
- If an SQL logic block contains a SELECT command, this command must be the only command in the SQL logic block.
- If an SQL logic block doesn't contain a SELECT command, it can contain up to eight SQL commands.
- Each command in an SQL logic block must be terminated with a semicolon (;).

---

## Another SQL Logic Block

The *training* application requires one further SQL logic block, the *commit* SQL logic block, which is called from the *product\_proc* process.

### SQL logic block - *commit*

The *commit* SQL logic block contains the single command:

```
COMMIT WORK;
```

#### To enter this command:

1. Define the the SQL logic block header.
2. Enter the details for the SQL logic block on the SQL logic block details screen.

Remember to terminate the command with a semicolon, and generate the SQL logic block after you create it.

HP ALLBASE/SQL requires you to issue an explicit COMMIT WORK to complete a transaction and make it permanent in the database. If you terminate a transaction or attempt to start a new transaction without issuing a commit work command, HP ALLBASE/SQL reverses any currently open transaction.

You can use an SQL logic block called from an HP ALLBASE/4GL logic block to issue the COMMIT WORK command.

---

## Summary

SQL logic blocks are used to pass SQL commands to HP ALLBASE/SQL. The important characteristics of SQL logic blocks are listed below.

- An SQL logic block comprises a header and the logic block details. You cannot complete the details for an SQL logic block until you have defined the header.
- If an SQL logic block contains a SELECT command it must be the only command in the logic block. An SQL logic block can contain up to eight SQL commands if it does not contain a SELECT command.
- SQL logic blocks must be generated to produce an executable form of the commands in the logic block.
- The commands in an SQL logic block are executed by the SQL command in a function or a process.
- To retrieve a row from an HP ALLBASE/SQL table, you can use a SELECT command in an SQL logic block to declare and open a cursor, and then use a FILE \*NEXT command in a function or process to retrieve the row into the HP ALLBASE/4GL file buffer.
- HP ALLBASE/SQL requires you to issue an explicit COMMIT WORK command to complete a transaction and make it permanent in the database.

You have completed all the logic you need for now. Complete the Self Test Questions for this chapter. Then you can start testing the application. Chapter 5, Lesson 9 explains the procedures for testing your application.

### 4-58 Defining Application Logic

---

## Self Test Questions

**Question 1.** What is an HP ALLBASE/4GL process?

**Question 2.** What is the difference between a process and a function?

**Question 3.** How many steps can an HP ALLBASE/4GL process or function contain?

**Question 4.** Why should a logic block be generated?

**Question 5.** What is the difference between the SHOW command and the SHOW \*REFRESH command?

**Question 6.**

### KSAM and HP TurboIMAGE/iX Developers Only

What does the following command do, and what does the asterisk (\*) at the end of the logic step refer to?

```
FILE *READ product *KEY = *
```

### HP ALLBASE/SQL Developers Only

What restrictions are placed on the usage of the SQL SELECT command within HP ALLBASE/4GL?

**Question 7.** Write the logic step to display a screen called *test*.

**Question 8.** List four ways of calling and executing a function.

---

## Answers

**Answer 1.** A process is similar to a program in a conventional language. It initializes the environment to a known state.

**Answer 2.** A function does not affect the current environment. When processing of a function is complete, processing returns to where the function was called from. When a process starts, HP ALLBASE/4GL initializes the environment to a known state. It closes all data files, clears all data buffers, and terminates any current non-background process, screen, function, report or decision table.

**Answer 3.** Logic blocks may contain a maximum of 30 lines.

**Answer 4.** When a logic block is generated, HP ALLBASE/4GL checks for the existence of data names, named records and files, and resolves pointers to these items. If you don't generate a logic block, these pointers cannot be resolved.

**Answer 5.** The SHOW command redisplay the screen with the contents of the screen buffer.

The SHOW \*REFRESH command performs the data movements specified and then redisplay the fields on the current screen.

**Answer 6.**

### **KSAM and HP TurboIMAGE/iX Developers Only**

This command reads a record from the file *product* where the key field matches the contents of the current screen field. The record is read into the file buffer. The \* refers to the current screen field.

### **HP ALLBASE/SQL Developers Only**

If you use a SELECT command in an SQL logic block, you cannot use any other command in the same logic block.

You cannot use the SELECT \* FROM form of the SELECT command, and you cannot use the INTO clause in a SELECT command.

**Answer 7.** SCREEN test

**Answer 8.** You can call and execute a function in the following ways. Your answer should include any four of the following:

## **4-60 Defining Application Logic**

- From a VISIT command in a logic block.
- As a function on a screen field.
- As an action item on a menu or decision table.
- From a function key.
- From a report.

This completes the preparation of the application. Now you can start testing it. The next lesson explains the procedures for testing your application.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Testing the Application

---

So far you have defined the data for your application, created the screens you need, and defined the supporting logic. Now you are ready to test whether your application will run. This chapter shows you how to use the *test* option and how to use the *Trace Mode* facility to check the step-by-step operation of the application. In addition, this chapter also introduces you to some useful utility functions.

To help you understand the logic used in HP ALLBASE/4GL applications, a separate section provides a detailed explanation of the HP ALLBASE/4GL screen processing logic. This section is for informational purposes and is not included as part of the lesson material. You may read it now, or refer to it later.

---

## Lesson 9 - Testing the Training Application

---

### Objectives

When you have completed this lesson, you will have learned how to:

- Use the *Test* option on the developer main menu to run the application you have just developed.
- Use the *Trace Mode* facility to examine the step by step operation of the application.

---

### Application Testing

You have now completed the development stages of a small, but workable application. The next step is to test it.

As a developer, you can test your application without leaving the *developer* application. The *Test* option on the developer main menu runs your application as it appears to the end user. When you test an application using the application testing option, you have access to some facilities that are not available to end users. These facilities can help you identify problems.

#### Any Problems?

Sometimes, things can go wrong when you test your application. The application may not behave as expected or it may even abort and return you to the developer main menu.

The remainder of this lesson assumes that nothing does go wrong. If things don't work as you expected, refer to "Handling Problems". It covers the most common problems that may occur.

If any problems you encounter are not explicitly described, refer to the *HP ALLBASE/4GL Developer Reference Manual* for further information.

#### 5-2 Testing the Application



**To test your application:**

1. Return to the developer main menu and select the *Test* option.

This loads a new HP ALLBASE/4GL environment and executes the initial action for the application. The initial action is either a *menu* or *process*. This is defined in the administrator definition for the application. The main menu for the application should appear.

2. Activate *Product Details*

This executes the *product\_proc* process. This process in turn displays the *product\_scrn* screen.

3. Enter data into the screen as follows:

<b>Action</b>	<b>Explanation</b>
Enter abc	HP ALLBASE/4GL alters the characters to upper-case. It also responds with a message to tell you that the data is less than the required minimum length. (Remember that you specified that this field must be six characters long.) This is a standard HP ALLBASE/4GL field validation.
Enter 123456	Before moving the cursor to the next field HP ALLBASE/4GL executes the <i>product_key_read</i> function associated with this field. This function is an <i>after</i> function because it is executed <i>after</i> the field is committed. The function reads the product file and searches for a record with a primary key that matches the product number just entered. This record does not exist, so HP ALLBASE/4GL displays a warning message. For the HP TurboIMAGE/XL based application, an HP TurboIMAGE/XL warning message is also displayed.

Action	Explanation
Press <b>(Return)</b>	The <i>Description</i> field has been defined as a required field. You cannot move to the next field if the <i>Description</i> field is blank. HP ALLBASE/4GL displays a message informing you that the field cannot be left blank.
Enter <b>test product</b>	The value for the product description is accepted. The cursor moves to the supplier number field when you press <b>(Return)</b> .
Press <b>(Shift)</b> and <b>(Tab)</b> . Then press <b>(Return)</b>	This is also a required field. However, you can leave it blank and move back to the previous field on the screen.
Press <b>(Tab)</b>	This moves the cursor back to the supplier number field.
Enter <b>1</b>	This value is accepted and the cursor moves to the lead time field after you press <b>(Return)</b>
Press <b>(Return)</b>	This is not a required field, so you can leave it blank by pressing either <b>(Return)</b> or <b>(Tab)</b> . You are prompted to press the <b>Commit Data</b> function key to complete the screen.
Press <b>Commit Data</b>	This terminates the screen and returns control to the process driving the screen. The process writes the new record to the file and then returns to the start, displaying the screen again.

---

## Using Trace Mode

When you first test your application it may not appear that all of the activity you have defined has actually taken place. Trace mode enables you to see it all.

HP ALLBASE/4GL has a trace mode to help you test your applications. It lets you see each step of your application as you test it, by displaying a line-by-line description of the logic being executed as the application runs.

### 5-4 Testing the Application

**To use trace mode:**

1. Press the **Previous Menu** function key to return to the main menu of the training application.
2. Then follow the next steps:

<b>Action</b>	<b>Explanation</b>
Press <b>System Keys</b>	A new function key set is displayed.
Press <b>More Keys</b>	Another set of function keys is displayed.
Press <b>Trace Mode</b>	This key turns the trace mode on. An asterisk appears in the function key label indicating that trace mode is now on. When you want to turn trace mode off again, simply press the function key again.

With trace mode on, you will now go through the procedure to add a product file entry again. This time you will see every logic block line echoed at the base of the screen as the application runs.

**To check the application activity:**

1. Activate the *Product Details* menu item.

The first line of the *product\_proc* process is displayed at the bottom of the screen in the following format:

```
P-product_proc: 1:MODE:MODE *WRITE product
```

or, for HP TurboIMAGE/XL based applications,

```
P-product_proc: 1:DM:DM IMAGE *MODE *MODLOCK :D-traindb
```

The colons (: ) separate the four parts of a trace statement. The four parts of a trace statement are:

- a. Logic block type and name. In trace mode, action prefixes are recognized, so P- indicates a process followed by the name of the process *product\_proc*.
- b. Step number within the *product\_proc* logic block. In the example above, this is step number 1.

**Testing the Application 5-5**

- c. Command name. In this case it is the MODE command (or the DM command for HP TurboIMAGE/XL based applications).
- d. The full command line; exactly what appears in the step, including the command name.

When the line has been executed, an end step message is displayed.

```
P-product_proc:1:END STEP
```

The next step, the SCREEN command, is displayed and executed. The *product\_scrn* screen is displayed.

2. Enter a product number.

After you press **Return**, the first line of the *product\_key\_read* function is displayed.

The logic for this function differs between data managers. As an example, the following should occur for KSAM based applications.

```
F-product_key_read: 1:FILE:FILE *READ product *KEY= *
```

This shows that HP ALLBASE/4GL is executing the function associated with this field after you entered data in the field. When the final step from the function finishes, the cursor moves to the next field and you can continue with data entry.

If you enter the number of a *product\_rcrd* record that already exists, the details of that record are displayed by the SHOW \*REFRESH command. If you are developing the HP ALLBASE/SQL based application, you cannot modify an existing record. If you try to modify a record, an error will occur.

3. Enter some details for each field, and then press the **Commit Data** function key. If you leave any required fields blank, HP ALLBASE/4GL displays an error message and returns you to the blank field. Otherwise you will exit from the screen processing and the following message is displayed:

```
P-product_proc: 2:END STEP: SCREEN product_scrn
```

This indicates that you have just completed the SCREEN command and have returned to the *product\_proc* process. The remainder of the process is executed (and several messages are displayed) and then returns you to the start of the process. After the record has been written to the file, you are

## 5-6 Testing the Application

returned to the start of the *product\_scrn* screen. The screen is not cleared at the completion of the SCREEN command. The screen display is only altered when a SCREEN command for a different screen is issued, or the user returns to a previous menu.

---

**Note**

Trace mode creates a file containing all the lines that are echoed to the screen while trace mode is active. This file defaults to a temporary file called HP4TRACE in your current logon group and account.

---

4. To exit from your application, press the **Previous Menu** function key twice from the *product\_scrn* screen, or once from the application *main* menu.

---

## Handling Problems

Occasionally, your application will not behave as you expect, or you will receive error messages while you're testing the application. The next few pages show you how to tackle some common problems.

### Problem: Items Not Found

The most common problem encountered when testing an application is discovering a reference to an undefined item. HP ALLBASE/4GL detects references to undefined items at different stages and the system response depends on the type of item that has not been defined.

Generating an item resolves references to certain application items. The nature and extent of the check applied to a referenced item depends on what type of item it is.

In general, the generate program doesn't attempt to resolve references to other generated items when generating an item. The exceptions are record layouts and calculated items.

The generate program attempts to resolve references to most non-generated items during generation. The exceptions are menus, help screens, and function keys.

The following table summarizes the various components, or items, that require generation, and the items that do not require generation. The table also indicates the items for which references are resolved during generation.

<b>Generated Items</b>	<b>Non-Generated Items</b>
Calculated items* Data screens Decision tables Functions Messages Processes Record layouts* Reports Window screens	Alphanumeric constants* Application titles* Field specifications* Files* Function keys Help screens Menus Numeric constants* Ranges* Scratch-pad field names* Validation tables* Variables*
* References to these items are resolved during generation.	

Since some references are not checked during generation, there is a possibility that you can successfully generate an application with unresolved references to the following items:

- Data screens.
- Decision tables.
- Function keys.

**5-8 Testing the Application**

- Functions.
- Help screens.
- Menus.
- Messages.
- Processes.
- Reports.
- Windows.

If the application encounters an unresolved reference at run time, you will receive an error message telling you that an item cannot be found. Usually processing can continue. If an undefined reference is critical to the logical flow of the application, you may not want processing to continue. In cases such as missing messages or help screens, the absence of the item may not have adverse effects on the logical flow of the application.

It's up to you to decide if you want to continue with testing the application when you run into this sort of situation. HP ALLBASE/4GL has a number of utilities to help you locate items in your application. Refer to Developer Utilities for a description of these utilities.

### **Problem: No Initial Action**

If HP ALLBASE/4GL can't find the initial action (as defined by the system administrator) for your application, processing cannot commence. If this happens, check the exact definition of the initial action for your application. You may need assistance from your system administrator to gain access to the administrator application. For details on accessing the application definition screen, refer to the *HP ALLBASE/4GL Developer Administration Manual*.

If you experience this problem you must ensure that the initial action is the correct type, and has the correct name. These details can be found in Appendix A. Remember, HP ALLBASE/4GL is case sensitive, so make sure you use the exact spelling of the initial action name.

---

## Lesson 10 - Using Developer Utilities

The *Utilities* option on the main menu lets you access the catalog display functions. You can use these functions to:

- Print a hard copy listing of your application, or selected portions of it.
- Copy an item that already exists in your application to another item. You can also copy an item from another application or version that exists in the same HP ALLBASE/4GL system.
- Delete any item that you have defined in your application.
- Display a catalog of the names of items defined in your application.

This item is also available from other menus in the developer.

Chapter 13 of the *HP ALLBASE/4GL Developer Reference Manual* includes additional utility programs that you may find useful. Those programs are not discussed in this training guide.

---

## Objectives

When you have completed this lesson you will have become familiar with a number of the utilities available in the developer that allow you to manipulate and check items within your application.

---

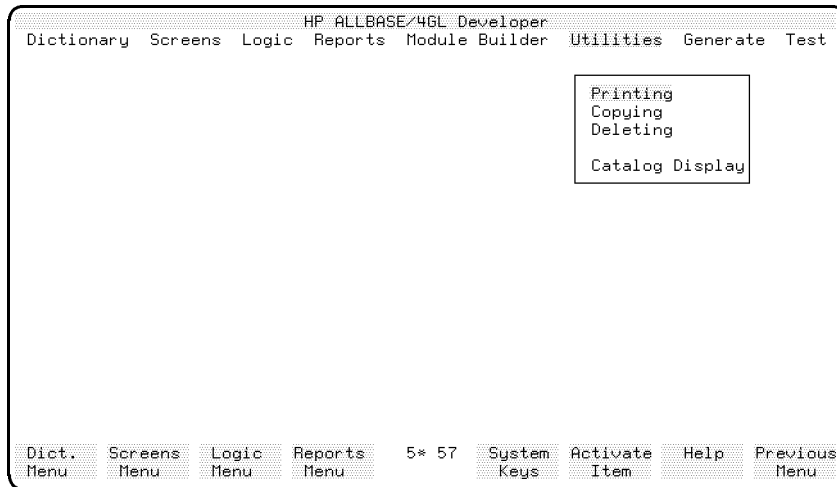
## The Utilities Menu

### To access the utilities menu:

1. From the main menu, select the *Utilities* option.

## 5-10 Testing the Application





### Utilities Menu

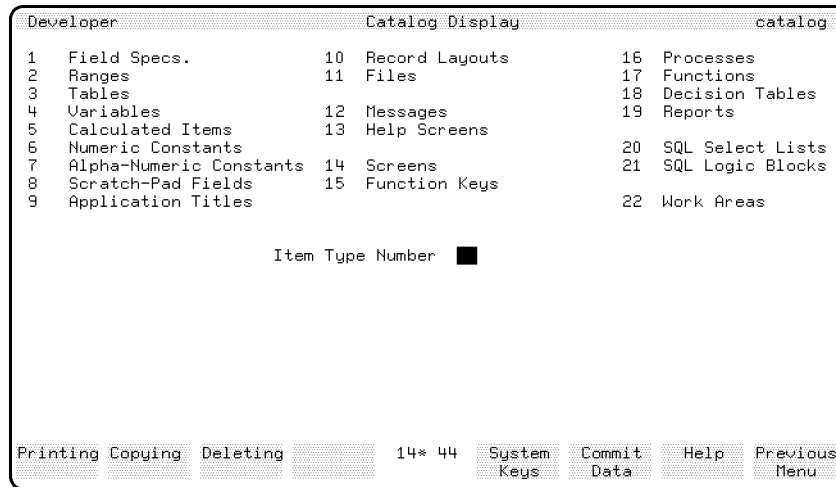
#### Displaying the Catalog

The catalog display utility allows you to look at the names of items defined in your application. This utility is available from most menus in the developer.

#### To activate the utility:

1. Select and activate *Catalog Display*.

The following figure will appear.



### Catalog Display Screen

There is only one data entry field on this screen. This is the number indicating the type of item names you want displayed.

2. Follow the steps shown here to display the names of the screens you have defined so far in your application.

Action	Explanation
Enter 14	This is the number corresponding to <i>Screens</i> . The upper part of the screen lists these items and numbers.
Press <b>Commit Data</b>	This displays the names of all the screens defined for this application.
Press <b>(Return)</b>	Depending on how many of these items you have defined for this application this has one of two effects. If there is more than one screen of names, another screen will be displayed. This process continues until all the names have been displayed. Once the last screen of names has been displayed you are returned to the catalog display screen.

## 5-12 Testing the Application

You can use the same method to display the name list of any items in your application.

With the catalog display utility, you can determine if the item you are looking for actually exists. If you have mistakenly created an item with an incorrect name, you can use the copy and delete utilities to correct the error. To rename an item, you must copy it to a new name and then delete the original.

Step through the item numbers and see the items you have created in your application.

### Copying Catalog Items

Return to the utilities menu and then select the *Copying* option. This allows you to copy an item that already exists in your application. You can also use this screen to copy an item from another application or version in the same HP ALLBASE/4GL system to an item in your application.

Developer		Catalog Copying		copying	
1	Field Specs.	10	Record Layouts	16	Processes
2	Ranges	11	Files	17	Functions
3	Tables			18	Decision Tables
4	Variables	12	Messages	19	Reports
5	Calculated Items	13	Help Screens		
6	Numeric Constants			20	SQL Select Lists
7	Alpha-Numeric Constants	14	Screens	21	SQL Logic Blocks
(n/a	Scratch-Pad Fields)	15	Function Keys		
9	Application Titles			22	Work Areas
Application to copy FROM		training			
Item Type Number					
Item Name to Copy FROM					
Item Name to Copy TO					
Description					
Last Modification:		Date	Time		
Printing	Deleting	Catalog	13* 48	System	Commit
		Display		Keys	Data
				Help	Previous
					Menu

### Catalog Copying Screen

#### Note



You are not required to make the following entries. However, you may if you wish to practice using the copying utility.

### To practice using the copying utility:

1. Check the name of the application or version you want to copy from. The display in the *Application to copy FROM* field defaults to the name of the application or version you're currently developing.

You would type over this default if you wanted to copy an item from a different application.

2. Enter the number that indicates the type of item you want to copy; enter 1 in the item type number field.
3. The next field is for the name of the item you want to copy; enter `product_no`
4. The last field is the name of the new item you want to create. If an item of this type with this name already exists, the copy is not allowed.

Enter `product_no_new`

5. To actually copy the item, press the `Commit Data` function key. When the copy is finished, HP ALLBASE/4GL prompts you for the name of the next item to copy.

Now *product\_no\_new* has the same definition as *product\_no*.

---

### Note



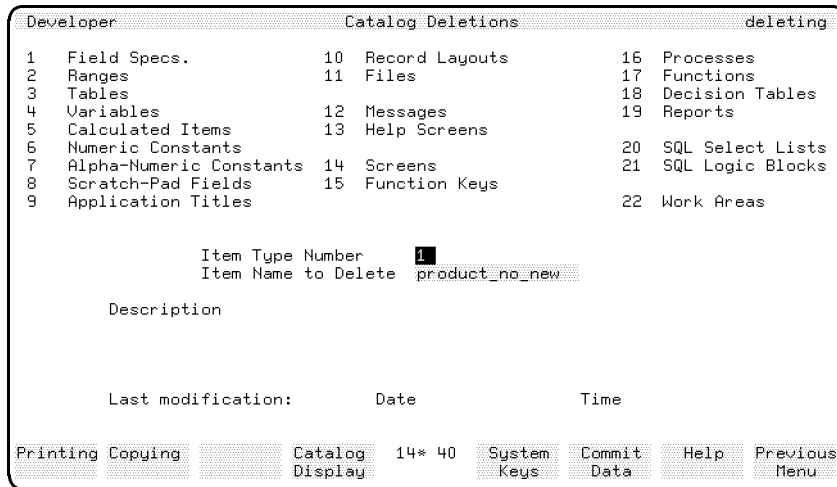
Since you can copy items from another application, you could set up a “library” application on your system to store a variety of standard items. For example, you may have some standard field specifications or functions that are used by a number of applications. Instead of creating the item each time it's needed, you can create the item once in the library application, and then copy the items as you require them.

---

### Deleting Catalog Items

Return to the utilities menu and select the *Deleting* option. This allows you to delete any item defined in an application.

### 5-14 Testing the Application



### Catalog Deletions Screen

#### Note



You are not required to make any entries here. If you did create the *product\_no\_new* field specification previously, then you can follow these steps to delete it.

#### To practice deleting an item:

#### Caution



This is a physical deletion of the item. You cannot retrieve an item after you delete it.

1. Once again, you must first enter the number indicating the type of item you want to delete.  
Enter 1
2. Then you can enter the name of the item to be deleted.  
Enter *product\_no\_new*
3. When you press **Return**, HP ALLBASE/4GL displays the description of the item if it has one.

4. Press the **Commit Data** function key to delete the item.

After you press the **Commit Data** function key, HP ALLBASE/4GL displays a message confirming that the item has been deleted.

## Printing Catalog Items

Return to the utilities menu and select the *Printing* option. This enables you to print a hard copy listing of your entire application, or selected portions of it.

```
Developer                               Catalog Printing                               printing
1  Field Specs.                            10 Record Layouts                            16 Processes
2  Ranges                                  11 Files                                       17 Functions
3  Tables                                   12 Messages                                   18 Decision Tables
4  Variables                               13 Help Screens                               19 Reports
5  Calculated Items                        14 Screens                                    20 SQL Select Lists
6  Numeric Constants                       15 Function Keys                             21 SQL Logic Blocks
7  Alpha-Numeric Constants
8  Scratch-Pad Fields
9  Application Titles
                                           22 Work Areas
Catalog Item Type Number                 1 (1 - 22 or A for All Item Types)
Short Index List or Full Index List      N (S/F/None)
Print Catalog Items Details               Y (Y/N)
All Entries or Selected Names             S (A/S)
                                           Selected Names
product_no    description
Copying      Deleting Catalog      19* 41      System      Commit      Help      Previous
              Display              Keys         Data        Menu
```

**Catalog Printing Screen**

---

### Note



You are not required to make any entries here, however you may wish to print a list of the items you have developed so far. If so, follow the instructions below.

---

## 5-16 Testing the Application

**To enter the field values:**

Field	Entry	Explanation
<b>Catalog Item Type Number</b>	1	Allows you to indicate the type of item you want printed. You can either enter the appropriate code 1 - 22 for the group of items you want, or enter <b>A</b> to specify that you want all item groups printed.
<b>Short Index List or Full Index List</b>	Accept <b>N</b>	You can print an index list of catalog item names with the text of the short description field or with the full description text. To list item names and their short description, enter <b>S</b> for <i>short index</i> . To list item names and their full text description, enter <b>F</b> for <i>full index</i> . Enter <b>N</b> if you don't want an index list of catalog item names.
<b>Print Catalog Item Details</b>	Accept <b>Y</b>	Enter <b>Y</b> if you want to print the detailed definition of each item. Enter <b>N</b> if you only want to print an index list.
<b>All Entries or Selected Names</b>	<b>S</b>	Here you can specify that you wish to have all the entries for the particular item group printed by entering <b>A</b> . If you enter <b>S</b> for selected entries you must complete the next fields.
<b>Selected Names</b>	product_no and description	<p>These fields allow you to enter the names of up to 16 items for the given type to be printed. If you enter any names in these fields, the entry in the field above automatically changes to <i>S</i>.</p> <p>If you don't want to enter the names of any items, press the <b>Commit Data</b> function key.</p>

**To print the items:**

1. Press the **Commit Data** function key to commence the printing.

If no items of the type you specify can be found, HP ALLBASE/4GL displays a message and no report is produced. If you have given a list of names of items to be printed, those that can be found are printed. A message is printed on the report if a requested item could not be found.

By default, the report is printed on the system line printer.

---

**Generating Items**

Sometimes your application may not proceed correctly even though you have correctly defined all the items. The most probable cause is that you haven't generated an item that requires generation.

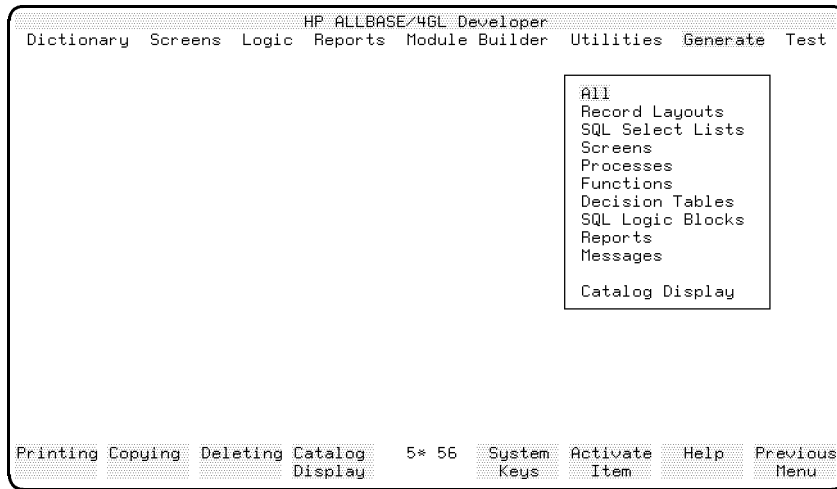
**To generate a single item:**

1. Go to the screen where you defined the item.
2. Call up the item.
3. Press the **Generate** function key.

The *Generate* menu allows you to generate groups of items in your application. Select the *Generate* option on the main menu to access this menu.

**5-18 Testing the Application**





### Generate Menu

---

#### Note



There is no need for you to actually do anything here, although you won't cause any damage if you do test the system.

---

The generate menu allows you to generate all the defined items within a particular group, or all the items within your application. If you press **Return** or the **Activate Item** function key, all the items of the selected type will be generated.

When the generation process commences, a number of messages are displayed. If necessary, the generate program is called. A message then advises you which type of items will be generated, and the generation for each item commences.

If each item generates without error, HP ALLBASE/4GL displays a confirmation message. However, if an error is encountered, a message appears detailing the errors. This display remains on the screen for about 30 seconds. Generation of the next item then commences. Generate will report up to 20 errors. If more than 20 errors are detected, the generation continues, but no more errors are reported.

---

**Note**

When you call generate, HP ALLBASE/4GL creates a file to log all generate errors. The file defaults to a temporary file called HPTGNERR in your login group and account. Each time an error is encountered by generate, the name of the item and the errors are listed to this file. This means that you can set a *generate all* going and then leave your terminal. When you return you can look at the file for any errors.

---

The usual reason for generating a group of items is to update the items to reflect a change in the definition of an item which is resolved during a generate. For example, if you modify the length of a field specification, the change would only be reflected in the screens referring to this specification after you generate the screens again.

---

**Summary**

This lesson introduced you to the utility functions available in the developer. These utilities allow you to:

- List the names of the items in an application.
- Copy items within an application, or copy items from other applications or versions.
- Delete items in an application.
- Print documentation about some or all of the items in an application.
- Generate all or part of an application.

If you feel comfortable with your application and your understanding of screen processing logic, go to the Self Test Questions at the end of the chapter.

Otherwise, turn the page for a discussion of Screen Processing that will aid you in developing applications using HP ALLBASE/4GL.

**5-20 Testing the Application**

---

## Understanding Screen Processing Logic

The HP ALLBASE/4GL screen processing logic controls:

- Information display on the screen.
- Data acceptance from fields on the screen.
- Execution of screen field functions.
- Movement of data to and from the screen field buffers and other data fields within HP ALLBASE/4GL.
- The sequence of screen field processing.

This section describes the three major components of the data screen processing cycle. These components are:

- Full data screen processing.
- Display field processing.
- Input field processing.

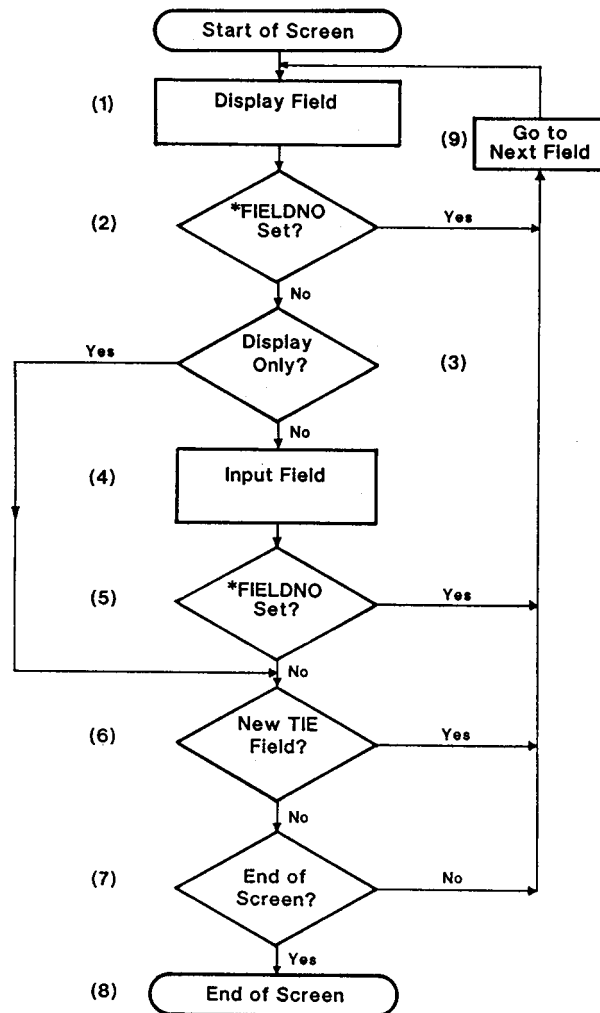
The descriptions of flow diagrams on the following pages will make it easier for you to understand HP ALLBASE/4GL screen processing logic.

---

## Full Data Screen Processing

This order of processing controls the way the cursor moves from field to field, and how the screen is completed.

The numbers in the following description refer to the step numbers in the diagrams.



1. When HP ALLBASE/4GL displays a data screen, the display field processing is invoked for the first screen field. (This processing is described later.)
2. HP ALLBASE/4GL tests the value in the \*FIELDNO communication area field. \*FIELDNO is a communication area field containing the number of

## 5-22 Testing the Application

the current screen field. This value can be set with the MOVE command to explicitly move a screen field number to \*FIELDNO. If \*FIELDNO has been set, control passes immediately to the field specified in \*FIELDNO. Otherwise processing continues normally.

3. The current field is tested to check if it allows data input. If it does, processing continues; otherwise it branches to step 5.
4. Input field processing is performed. (This is described later.)
5. At the end of the input field processing, HP ALLBASE/4GL tests the contents of \*FIELDNO again. If the \*FIELDNO value has been changed during input field processing, control passes to the field indicated by the new value in \*FIELDNO. Otherwise processing continues normally.
6. This step tests the communication area field \*NEWTIE to determine if it has been altered during input field processing. \*NEWTIE is a communication area field that contains the number of the next field to be processed. If
7. NEWTIE has been altered, control passes to the screen field indicated by the value in \*NEWTIE. Otherwise processing continues normally.

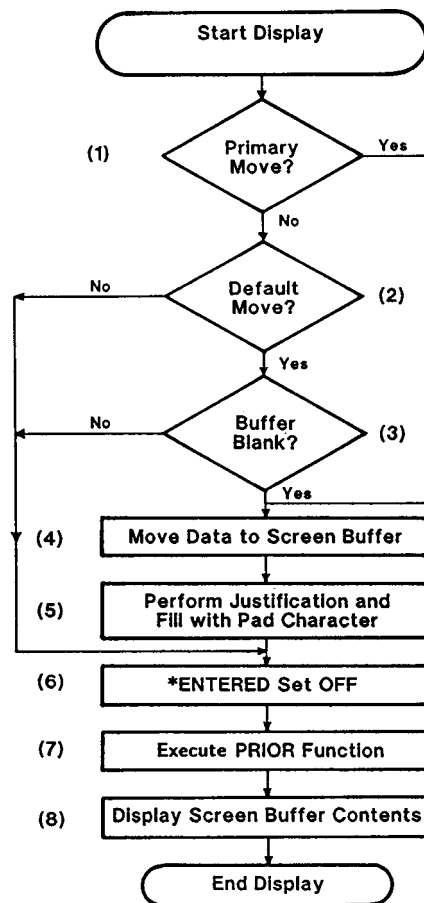
The value in \*NEWTIE can be set by moving a new value to it, or by the TIE logic command. The TIE command takes a screen field name as its argument and moves its number to

8. NEWTIE.
9. Next HP ALLBASE/4GL determines if there are any more fields on the screen. If so, control passes to the next screen field.
10. At the end of the screen, one of two things can occur. If the screen has input fields, the cursor remains on the last input field. If the screen doesn't have any input fields, the cursor is positioned at the base of the screen. In both cases, the system waits for the user to press a valid function key.
11. This step resets system values such as \*FIELDNO and \*NEWTIE and passes control to the next field.

---

## Display Field Processing

HP ALLBASE/4GL performs display field processing for every field (display and input) on the screen. The display processing for a field occurs when control first passes to the field.



1. HP ALLBASE/4GL determines if a primary data movement field is specified for this screen field. If one is specified, processing branches to step 4.

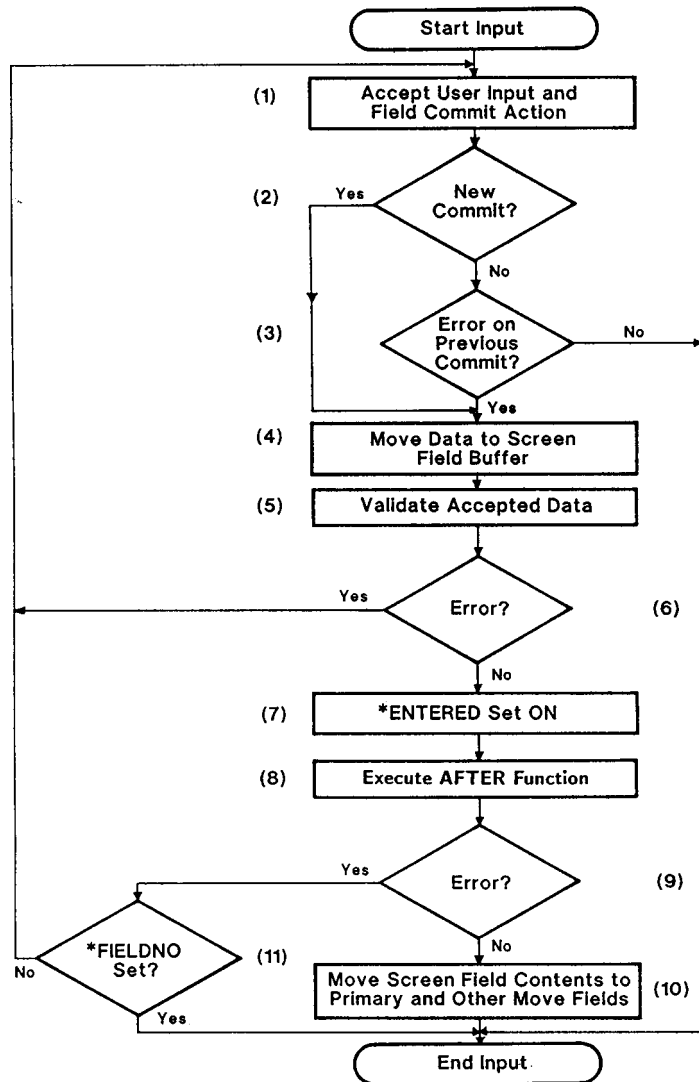
### 5-24 Testing the Application

2. This step checks if a default data movement field is specified. If not, processing branches to step 6.
3. The current screen field's buffer is tested. If it contains data, processing branches to step 6. HP ALLBASE/4GL only moves data from the default data movement field to the screen field buffer if the screen field buffer is blank.
4. HP ALLBASE/4GL moves data from the primary data movement field (or the default data movement field) to the screen field buffer.
5. The screen field buffer contents are justified as required and filled, if necessary, with the pad character. They are not immediately displayed on the screen.
6. The switch \*ENTERED is set off.
7. If there is a prior function assigned to the field, (that is, the function has been defined to be executed either prior to or, both prior and after data entry), then it is executed now. This function can test \*ENTERED to determine when it is being called.
8. Finally, the contents of the screen buffer are displayed in the screen field.  
Display field processing is now complete and control returns to the full data screen processing logic described earlier.

---

## Input Field Processing

HP ALLBASE/4GL performs input field processing for input fields after display processing for the field has been completed.



1. When HP ALLBASE/4GL first enters this processing sub-system, it places the cursor in the current field and waits for user input. This is the only

## 5-26 Testing the Application



time that HP ALLBASE/4GL accepts any form of user input, including function keys or cursor movement requests.

A field commit action indicates that data entry for the field is complete. Some actions that constitute a field commit are:

- Moving the cursor off the field by:
  - Pressing **Return**,
  - Pressing **Tab** or **Shift** + **Tab** followed by **Return**,
  - Moving the cursor to another field by pressing **cursor up** **cursor down** or **cursor home** followed by **Return**.
- Pressing the **Commit Data** function key.

---

**Note**

In general, pressing a function key other than **Commit Data** does not initiate a field commit.

---

There are only two cases in which a field commit occurs as a result of pressing a function key:

- If the action behind the key initiates a commit data action, or
- If the action is a function which moves a new value to \*FIELDNO, a field commit occurs on return to the screen field.

Once HP ALLBASE/4GL accepts the field commit, the remainder of the input field processing is executed.

2. HP ALLBASE/4GL determines if the field commit is a new commit action for this field. A new field commit is recognized for the following situations:
  - The data in the screen field has altered since the last commit action for the field.
  - A different numbered input field has been processed since this field was last committed.
3. If the field commit is not a new commit action, HP ALLBASE/4GL checks to see if an error was generated by the previous commit action. If an error

was generated on the previous commit, this commit action is accepted. Otherwise input processing for the field is terminated.

4. The data input in the screen field is accepted into the screen buffer.
5. HP ALLBASE/4GL now validates the screen field buffer contents. Firstly, the following automatic system tests are performed:
  - **Field Requirements.** Validation fails under the following circumstances:
    - The field is blank and it is defined as a required field,
    - Less than the defined minimum number of characters for the field have been entered.
  - **Data Type Requirements.** HP ALLBASE/4GL performs the following data type validation:
    - Correct data type for edit code,
    - If a date field, validate the date format,
    - If numeric, validate number of decimal places,
    - If signed numeric, validate sign.

If any of these criteria are not met, HP ALLBASE/4GL generates an error signal, displays a standard system error message, and terminates the validation.

If the input field data passes the system validation tests, any developer specified validation tests (if specified) are applied. These tests are as follows:

- Value within a specified range.
- Value in a specified table.

If either of these tests fail, HP ALLBASE/4GL displays the developer defined message assigned to the range or table. If there is no message associated with the range or table, HP ALLBASE/4GL displays a default system error message.

Error messages are one of five different types of messages. They are the most important type of message in terms of their effect on input field processing logic. They provide an automatic means of ensuring that the

## 5-28 Testing the Application

user enters valid data in a field. You will define a number of messages in later lessons.

6. If the validation tests of the screen field data initiate an error message, processing returns to step 1 and the user must re-enter the field data.
7. When the field data has been accepted and validated, the switch \*ENTERED is set on. This allows you to determine whether the function is being performed prior to, or after, data entry for the field.
8. The function, if specified as an after function, or as both a prior and after function, is executed.
9. The function, through its processing, may initiate an error message. If so, processing branches to step 11.
10. HP ALLBASE/4GL now performs the automatic data movement for the field. The contents of the screen buffer are moved to the primary data movement field, and the other data movement fields. Processing of the field is complete.
11. If the after function generates an error message, HP ALLBASE/4GL tests the value in the \*FIELDNO communication area field. If this value has not been modified by the function, processing returns to the beginning of the field (Step 1). However, if the function has modified the value in \*FIELDNO, HP ALLBASE/4GL restores the contents of the field to the condition that existed at the start of the input processing, and then terminates processing of the field.

---

## Summary

It's important to understand the actions that occur during data screen processing. A number of system items are available to help you make full use of data screens, and make them operate efficiently for your users. These system items are summarized below.

- **\*FIELDNO.** A communication area field containing the number of the current screen field. You can alter this value by moving a new number to it with the MOVE logic command. Moving a value to \*FIELDNO alters the

screen field processing sequence, allowing you to branch to a field depending on the value of another.

- **\*NEWTIE.** A communication area field that contains the number of the next field to be processed. You can alter this value manually using the TIE command, or by moving a new value to it with the MOVE command. It differs from \*FIELDNO in that it only takes effect once all normal input processing for the field has finished.
- **\*ENTERED.** A read-only switch that enables you to determine when a screen field function is being executed. \*ENTERED is off before data entry, and on after successful data entry.
- **ERROR Messages.** If an ERROR message is generated by a data input validation test or by an after function, HP ALLBASE/4GL automatically requests the user to correct the data entry and commit the field again. This prevents the user committing invalid data.

## Data Movement

When HP ALLBASE/4GL first displays a field, it moves data to the internal screen field buffer before it executes the field function. Data is not displayed on the screen until the function finishes. This means you can refer to the screen buffer contents and modify them with the function. The modified data is displayed upon completion of the function.

HP ALLBASE/4GL executes the after function for a field before the data movement for the field occurs. This means that the after function cannot rely on having the values available in data movement destination fields. However, the after function can access the value in the internal screen field buffer, and if required, modify this value before the data movement for the screen field occurs.

If any automatic data validation, or data validation in an after function, results in an error, the normal data movement for the screen field does not occur.

## 5-30 Testing the Application

---

## Self Test Questions

**Question 1.** How does HP ALLBASE/4GL know what is the first action to be executed when an application is run?

**Question 2.** Where can you locate the **Trace Mode** function key?

**Question 3.** Can you move to another field if the current field is a *Required* field and is blank?

**Question 4.** Which of the following items in HP ALLBASE/4GL need to be generated?

- Files
- Data Screens
- Menus
- Record Layouts

**Question 5.** Which of the following items in HP ALLBASE/4GL do NOT need to be generated?

- Application Titles
- Field Specifications
- Messages
- Processes

**Question 6.** What is \*FIELDNO and what does it contain?

**Question 7.** Can the status of the \*ENTERED switch be changed programmatically?

**Question 8.** When HP ALLBASE/4GL displays a field, which data movements are valid, and will they (if specified) always be performed?

---

## Answers

**Answer 1.** The first action executed is the *Initial Action* that is specified by the HP ALLBASE/4GL administrator when the application is defined.

**Answer 2.** The **Trace Mode** function key is only available within application testing. To locate the key, press the **System Keys** function key, and then the **More Keys** function key.

**Answer 3.** Yes, you can move to previous fields from a required field. However, you cannot move to any fields after the required field until the required field is completed.

**Answer 4.** Data screens and record layouts must be generated.

**Answer 5.** Application titles and field specifications do not need to be generated.

**Answer 6.** \*FIELDNO is an alterable communication area field. It contains the number of the current screen field.

**Answer 7.** No, the status of the \*ENTERED switch cannot be changed programmatically.

**Answer 8.** During display field processing, only Primary data movements and Default data movements are performed. The Primary move is always performed, but the Default move is only performed if the screen field is blank.

## HP ALLBASE/4GL Reports

---

Now that you have tested your application, you are ready for the final phase of development: creating reports. The lessons in this chapter step you through the definition and creation of a typical report. The report you are about to create enables you to look at the entire contents of the product file that you have created.

The lessons in this chapter cover three areas:

- Defining a report.
- Painting a report image.
- Testing a report.

HP ALLBASE/4GL lets you design reports that are printed on a printer or displayed on the user's terminal. A report painter enables you to quickly create an image of a report. HP ALLBASE/4GL also allows you to process data before it is included in a report.

---

### What is an HP ALLBASE/4GL Report?

Some of the concepts in this chapter may be new to you if you haven't used a report generator before. An HP ALLBASE/4GL report is not created or processed in a physical line by line manner as most reports are. An HP ALLBASE/4GL report is an arrangement of print lines defined by you, and printed in an order determined by the report processor based on your definition of the report.

## Format of a Report

The data included in a report is taken from a data file called the report's *primary file*. Data can also be included from other files as well. This is done by using file linkages to the other files. The report processor reads through the primary file sequentially and creates a temporary file to produce the actual report. The temporary file contains the records that meet the defined selection criteria, and they are sorted according to the specified sort sequence. Once the selection and sorting process of records from the primary file is complete, the report processor sequentially prints the records from the temporary file.

Three important concepts in HP ALLBASE/4GL reports are:

- Line groups.
- Line numbers.
- Totalling facilities.

The following descriptions provide an overview of these concepts.

### Line Groups

The format of an HP ALLBASE/4GL report is defined by a series of different line groups. A line group is a logical entity that the report processor prints at a certain stage in the report printing process. HP ALLBASE/4GL allows you to use a number of different line groups. They are identified as follows:

Name	Description
P1	Top of page headings.
C1	Column headings.
B1	Bottom of page footings.
D1 to D9	Detail lines.
E1 to E9	Extra lines.
H1 to H8	Subheading lines.
T1 to T8	Subtotal lines.
TF	Final total line.

## 6-2 HP ALLBASE/4GL Reports



As their descriptions suggest, HP ALLBASE/4GL processes each line group at a different stage of the report. For example, HP ALLBASE/4GL prints the P1 line group at the start of a new page, followed immediately by the C1 line group. When the report concludes, HP ALLBASE/4GL prints the TF line group. A line group that is not defined for a report will not be printed. Only the D1 line group must be defined for each report. HP ALLBASE/4GL prints a D1 line group for each record selected for reporting.

### Line Numbers

You may be wondering why line groups are named *groups*. A line group may, in most cases, consist of up to 99 physical print lines. The following table summarizes the number of lines that may be used within each line group.

Line Group	Line Number Range
P1	1 to 99
C1	1 to 3
B1	1 to 9
D1 to D9	1 to 99
E1 to E9	1 to 99
H1 to H8	1 to 99
T1 to T8	1 to 99
TF	1 to 99

Each time the report processor prints a line group, all of the line numbers defined for that line group are printed. They are treated as one logical entity.

### Totalling Facilities

Most reports usually require some type of totalling. With HP ALLBASE/4GL reports there are a number of automatic totalling facilities available. The totals are maintained in a number of communication area fields. These are as follows:

Name	Description
*TOTALS(1) - (16)	Totals fields 1-16 allow you to total values down your report. You can accumulate the values in one field on many lines.
*CROSS(1) - (5)	Cross fields 1-5 allow you to total values across a physical line. You can accumulate the values from several fields that appear on one line.
*COUNT(1) - (5)	Count fields 1-5 will count the number of times that a specified print line has been printed.

With this basic outline of a report, you can now create a report for the *product* file.

---

## The Product Report

The report that you are about to create lists all the records in the product file.

The following sample printout gives you an indication of what your first report will look like. This sample also shows the line numbers, discussed previously, associated with each line printed.

```

P1.01:                Product File Details                Page: 0001
P1.02:                Date: 08/08/89

C1.01:  Product      Description                Supplier  Lead Time

D1.01:  XY1000
D1.02:                Blue pens,in batches of 20.    122      02 Days

D1.01:  XY9123
D1.02:                A4 Letterhead paper, 1 ream.    49       10 Days

TF.01:  Total Number of Products    2

```

The two line page heading contains the date of the report, a report title, and the current page number. A column heading line immediately under the page heading identifies the columns of data to be printed. The body of the report

### 6-4 HP ALLBASE/4GL Reports

contains the details of each product, printed over a maximum of two physical print lines for each product.

At the end of the report, a single line summary shows how many *product\_rcrd* records have been included in the report.

---

## Lesson 11 - Defining the Report

---

### Objectives

When you have completed this lesson you will have learned how to:

- Define a report header
- Define report line headers
- Paint a report image

To define a report, you must first define the report header. Then you define the remaining details of the report in any sequence. The screens you learn to use are:

- **The report header screen.** Allows you to define the operational environment of the report.
- **The report line header screen.** Allows you to define the characteristics of the lines that make up the report.
- **The report painter.** Allows you to paint an image of the various report lines you have defined.
- **The report sorting screen.** Allows you to define the order in which the records from the file are to be printed.
- **The selection criteria screen.** Allows you to define a number of selection criteria which must be met for a record to be included in the report.
- **The file linkages screen.** Allows you to specify when, and from which files, extra information is extracted while the report is being produced.

In the lessons in this chapter you will use the first three of these items. The others are optional. To keep this report simple, the optional items are not used.

---

## Task 1 - Creating the Report Header

The first item you must define for every report is the report header.

### Menu Path

From the main menu:

1. Select the *Reports* option
2. Choose *Header*

### Screen Description

This screen allows you to define the basic operational specifications of the report. These specifications include the name of the report, the primary file for the report, the type of stationery for the report, the printer to be used, and a brief description of the report.

Developer	Report Header	report_header
Report Name	product_rept	Secured <input checked="" type="checkbox"/> (Y/N)
Report File Designator	prod	
Primary File[.Record]	product	Index 1
Printer	D (F/L/D)	
Type of Stationery		Number of Copies 1
Characters per Line (maximum)	80	Actual Page Size in Lines 22
Print Lines Used per Page	22	Formfeed Skip to Next Page N (Y/N)
Start of Report Function Name		
End of Report Function Name		
Description	product_rept	
AUTHOR:	developr	
	This is a report to the screen of product data from the	
	product data file in the training application.	
Last Modification:	Date 4/13/92	Time 13:31:34
Sorting	Record	Line
Select	Header	Generate
	Report	3* 73
	System	Commit
	Keys	Data
	Help	Previous
	Menu	

**Report Header Screen**

## 6-6 HP ALLBASE/4GL Reports

**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Report Name</b>	<code>product_rept</code>	The name of the report you are creating.
<b>Secured</b>	Accept the default.	This specifies whether this report is secured against an unauthorized developer changing the definition.
<b>Report File Designator</b>	Accept the default of <code>prod</code> .	This is the name of the MPE/iX formal file designator for the report output file. You can use an MPE/iX file equation to direct the report to a specific device, or to a disk file. By default, the report is directed to the device class LP.
<b>Primary File[.Record]</b>	<code>product</code>	The name of the data file that is the primary file for this report. In this particular report, all the records in this file are printed.
<b>Index</b>	Accept the default of 1.	The index used to read the file. Since this report is not sorted, it also determines the order in which the records are printed. In this report, the file is printed in product number order.
<b>Printer</b>	D	Specifies whether the report is printed to a formal file designator (F), a local printer attached to your terminal (L), displayed on your terminal screen (D), or printed at printer 1, 2, 3 or 4. These four printers are defined by your HP ALLBASE/4GL administrator.
<b>Type of Stationery</b>	Leave this field blank.	Specifies the stationery type that should be loaded on the printer for this report. If the specified stationery is not loaded on the printer, MPE/iX requests the system operator to change the paper to the correct type.

Field	Entry	Explanation
<b>Number of Copies</b>	Accept the default of 1.	You can print multiple copies of the same report by specifying a value greater than 1 in this field.
<b>Characters per Line (maximum)</b>	Accept the default value 80	Specifies the maximum allowable report line width. The width you specify also determines the maximum width that the report painter allows you to use when you create an image of this report.
<b>Actual Page Size in Lines</b>	Accept the default of 22.	Specifies the physical form length of the paper for the report.
<b>Print Lines Used per Page</b>	Accept the default of 22.	The logical page size. That is, the maximum number of lines that can be printed on each physical page.
<b>Formfeed Skip to Next Page</b>	Accept the default of <b>N</b> .	Specifies the type of form feeding control that is used. If you use non-standard size stationery, you may need to use linefeeds to advance to the next page. Entering <b>Y</b> specifies a formfeed, and <b>N</b> specifies multiple linefeeds.
<b>Start of Report Function Name</b>	If you are developing the HP ALLBASE/SQL based application, enter <b>sel_product</b> . Otherwise, leave this field blank.	This is the name of a function that is executed before HP ALLBASE/4GL starts producing the report. This function can display a screen to obtain user input for parameters that are used within the report, or data that is used in the report.  To produce a report for an HP ALLBASE/SQL database, you must declare and open a cursor using the SELECT command in an SQL logic block. The start of report function <i>sel_product</i> will contain an SQL command to call an appropriate SQL logic block. You will create this after creating the report header.

## 6-8 HP ALLBASE/4GL Reports

Field	Entry	Explanation
<b>End of Report Function Name</b>	Leave blank.	The name of a function to be executed after the report is produced. You could use this function to update data files as a result of totals obtained during the generation of the report.
<b>Description</b>	Enter a suitable description.	

**To complete and commit the screen:**

1. When all the entries on this screen are correct, press the **Commit Data** function key to create the report header. The report header is now complete and you can go on to define the remainder of the report.

**The Next Step for KSAM and TurboIMAGE/iX Applications**

For the KSAM based and HP TurboIMAGE/iX based application, the next step is to define the print lines used for this report.

1. Press the **Previous Menu** function key to return to the reports menu. Then turn to “Task 2 - Creating the Report Line Header.”

**The Next Step for SQL Based Applications**

If you are developing the HP ALLBASE/SQL based application, you must first create the *sel\_product* function and the SQL logic block that it calls. Follow the procedure below.

**Start of Report Function**

**Function *sel\_product***

1. Create and generate the *sel\_product* function.

The *sel\_product* function reads as follows:

```

1 SQL select_product
2 EXIT

```

This function simply calls the SQL logic block *select\_product*. Note that this function does **not** contain a FILE \*NEXT command. The report generator performs the equivalent operation to a FILE \*NEXT command when it accesses the primary file for the report. In this case, the primary file for the report is the active set for the cursor defined by the SQL block *select\_product*.

2. Create and generate the SQL Logic Block *select\_product*.

The *select\_product* SQL logic block contains the following command:

```
SELECT :product FROM sqlgrp.product
ORDER BY product_no;
```

This SELECT command contains an ORDER BY clause. This clause ensures that records are retrieved in the required order for the report.

---

## Task 2—Creating the Report Line Header

Before you can actually paint the report image, you must define the lines that are used in the report. This report uses a total of eight print lines from four of the available line groups.

### Defining the Report Lines

#### Menu Path

1. From the main menu, select the *Reports* option.
2. Choose *Line Header*.

#### Screen Description

You use this screen to specify the line groups used by the report, and define the way in which each of the print lines is handled. This screen allows you to specify the following details for each line:

- How many blank lines to leave before or after printing.
- Whether the line is always printed.

### 6-10 HP ALLBASE/4GL Reports



- Which count field to use to count the number of times the line is printed.
- The names of functions to be executed before and after each line is printed.

Developer		Report Line Header		report_line_head			
Report Name	product_rept	Secured	N	(Y/N)			
Line Group	P1	(P1/C1/B1/Hn/Dn/En/Tn/TF)					
Line Number	01						
Action	A	(A/C/D)					
Skip Lines/Page:	Before Print	0	(0-9/P)				
	After Print	1	(0-9/P)				
Suppress Line If:	All Values Are Zero	N	(Y/N)				
	All Values Are Blank	N	(Y/N)				
Count Lines Printed Into Counter Number			(1-5,Blank)				
Underline Numbers			(P/A/B)				
Before Print Function							
After Print Function							
Grp #	Act	Skip	Suppress	Count	Underline	Function	Function
Header	Record	File	Generate	6* 23	System	Commit	Help
Select	Linkage	Report	Keys	Data	Menu		

### Report Line Header Screen

#### To enter the field values:

The characteristics of each line are defined one line at a time. The eight lines to be printed are P1.01, P1.02, C1.01, D1.01 through D1.04, and TF.01.

Field	Entry	Explanation
<b>Report Name</b>	Accept the default value of <b>product_rept.</b>	The name of the report for which you are about to define the lines. The name in this field defaults to the last report name used in any of the report development screens. When you commit this field, HP ALLBASE/4GL skips the secured field.
<b>Line Group</b>	P1	The line group identifier. The <i>P1</i> group is the page header group and it specifies the lines that HP ALLBASE/4GL prints at the start of every new page.

Field	Entry	Explanation
<b>Line Number</b>	Accept the default of 01.	The line number specifier within the line group that you just entered above. The lines are printed on the report in ascending numeric order. The combined identifier of <i>line group.line number</i> is known as a <i>print line</i> .
<b>Action</b>	Accept the default.	This action code is used in the same manner as the other action codes, that is: <i>A</i> = Add, <i>C</i> = Change, and <i>D</i> = Delete. The action defaults to an appropriate code that depends on whether the print line already exists, or you are defining a new print line.
<b>Skip Lines/Page</b>		The next two fields determine the line spacing used for this particular print line.
<b>Before Print</b>	Accept the default of 0.	This indicates that no line feed characters are issued before this line is printed. You can enter a value between 1 and 9 to specify a number of line feeds, or you can enter P to force a page break before this line is printed.
<b>After Print</b>	Accept the default of 1.	Indicates that one line feed character is issued after this line is printed. You can skip 1 to 9 lines after this line is printed, or force a page break.  <b>Note:</b> The print line buffer is only cleared after it is printed. If you specify that a line has 0 lines skipped after printing, the line is not printed and the buffer contents remain intact. If the next line printed skips 0 lines before printing, the data is merged over the first line and then printed.
<b>Suppress Line If:</b>		The next two fields allow you to specify that a line is only printed if it contains non-blank or non-zero data.

## 6-12 HP ALLBASE/4GL Reports

Field	Entry	Explanation
<b>All Values are Zero</b>	Accept the default of <b>N</b> .	When set to <i>Y</i> , this suppresses the printing of a line if all the numeric fields in the line are zero.
<b>All Values are Blank</b>	Accept the default of <b>N</b> .	When set to <i>Y</i> , this suppresses the printing of a line if all the alphanumeric fields in the line are blank.  <b>Note:</b> There are two sets of conditions for these tests. The tests are relevant where there are only numeric or only alphanumeric fields on the print line. However, if the line contains both field types, all numeric fields must contain zeros <b>and</b> all alphanumeric fields must be blank before printing the line is suppressed.
<b>Count Lines Printed into Counter Number</b>	Leave blank.	Indicates which communication area *COUNT field (*COUNT(1) to *COUNT(5)) is incremented each time this particular print line is printed. Leaving this entry blank specifies that you don't require automatic line counting.
<b>Underline Numbers</b>	Leave blank.	Indicates whether the numeric fields on the print line are to be overlined (P = Prior), underlined (A = After), or both (B = Both). A blank entry specifies that no underlining is done.  The underline or overline is printed on a separate print line. This line is counted as a regular line as the system tests to see if a new page is necessary. The data and its underlining or overlining always appear on the same page as it is treated as a single entity at print time.
<b>Before Print Function</b>	Leave blank.	The name of a function that is executed before the print line is printed. This function can perform data formatting, or extract data from another source to be included in the print line.

Field	Entry	Explanation
<b>After Print Function</b>	Leave blank.	<p>The name of a function that is executed after the print line is printed. This function could possibly initiate the printing of extra lines on the report. These extra lines could be based on the line just printed.</p> <p><b>Note:</b> The functions that are executed before or after printing a print line can set two switches. These are the *BYPASS and *ENDLINE switches. *BYPASS controls the printing of each print line and bypasses the current print line when set on. *ENDLINE controls the line group, and terminates the processing of the line group when set on. This allows you to perform more complex selection and validation of data that is to be printed.</p>

**To complete and commit the screen:**

1. Press the **Commit Data** function key to commit the definition of the line group.

HP ALLBASE/4GL displays a summary of the line at the base of the screen and then returns the cursor to the *Line Number* field. By default, the system expects you to define another print line in the same group.

2. If you want to define another line group, go back to the *Line Group* field. You can now proceed to define the remaining seven lines in this report.

**To complete the remaining lines:**

1. Enter the values required for each of the remaining lines of the report. Only those fields that require an entry other than the default are shown here.
2. Accept the default value for the other fields.
3. Press the **Commit Data** function key when you have completed the details for each line.

**6-14 HP ALLBASE/4GL Reports**

### Page Heading Line 2

Line Number	02
Skip Lines After Print	2

### Column Heading Line 1

Line Group	C1
Line Number	01

### Detail Line 1

Line Group	D1
Line Number	01
Skip Lines Before Print	1
Count Lines Printed ...	1

---

### Note



The total number of products included in the report is printed at the end of the report. Each time this line is printed, \*COUNT(1) is incremented to count the number of records that have been printed.

---

### Detail Line 2

Line Number	02
-------------	----

### Final Total Line 1

Line Group	TF
Line Number	01
Skip Lines before Print	3

This completes the definition of the characteristics for each line. You can now go on to paint the image of each line using the report painter.

---

## Summary

In this lesson you defined a report header and the line headers for the report. If you are developing the HP ALLBASE/SQL based application, you also created a start-of-report function and an SQL logic block.

The report header defines the operational environment for a report. It includes information such as the name of the primary file for the report, the printer to be used for the report, and the size and type of stationery for the report.

The report line headers define some details for the print lines that are used on a report. For each line, you can specify details such as the before print and after print line spacing, whether lines are suppressed if they are blank, whether a count is kept of the number of times the line is printed.

The reports menu does include some other screens for defining report sorting, record selection criteria, and file linkages. These screens are optional, and these additional definitions are not required for the *training* application report.

In the next lesson, you will use the report painter to complete the definition of this report.

---

## Lesson 12 - The Report Painter

---

### Objectives

When you have completed this lesson, you will have learned how to use the report painter to create the image of the *product\_rept* report.

---

### Painting a Report

The report painter is a program that is similar to the screen painter. In much the same way as painting a screen, you use the report painter to interactively create the layout of the data and text fields on a report.

As a report can be wider and longer than a standard terminal, the report painter offers you a window that you can move around over the report.

You can create two different types of fields on a report: text items and data fields. Text items are created by simply typing them in where they are needed.

A data field is where you show the data being reported. Typically the data comes from a file field, although it can come from any HP ALLBASE/4GL data item. You create a data field manually by painting its image on the report line, or automatically by referring to a dictionary field specification name.

The report painter also allows you to specify the source of the data that is printed in the report output fields.

#### Menu Path

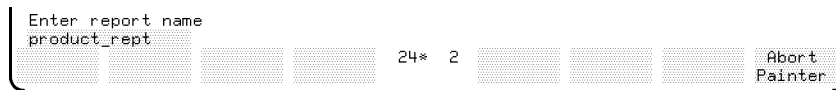
#### To access the report painter screen:

1. From the main menu, select the *Reports* option.
2. Choose *Painter*.

**To start the report painter:**

1. Activate the *Painter* menu item on the reports menu.

When the report painter starts, you will see this prompt at the bottom of the screen:

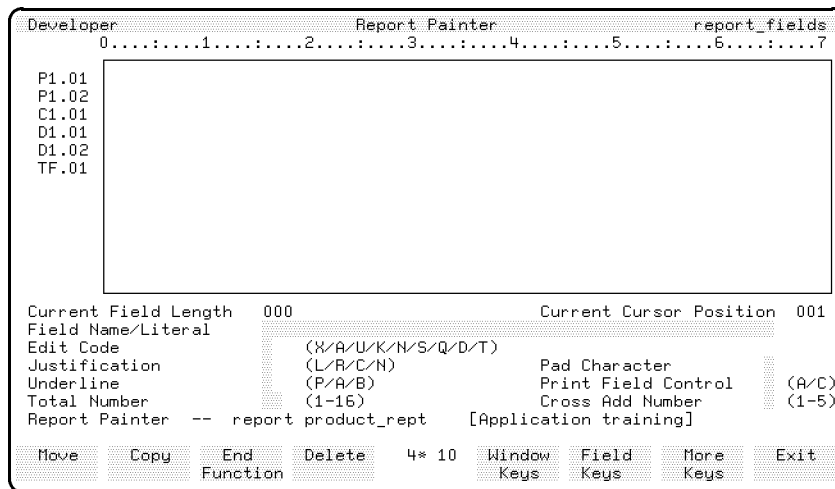


**Report Painter Prompt**

2. Accept the default or enter the name of the report that you want to paint.

The name of the report which you have been defining so far will be displayed as the default entry. In this case it should be *product\_rept*.

3. Press **Return**.



**Report Painter Screen**

**6-18 HP ALLBASE/4GL Reports**



## Screen Description

The report painter screen is divided into two regions. The upper region displays the window into the report. The lower region displays the details of the current report field.

A ruler line across the top of the report window shows your current position on the report page.

There are two important display fields at the top of this lower region. *Current Field Length* shows the length of the current field on the report line. The current field is the field currently occupied by the cursor. When you press **Return**, the report painter highlights the current field and updates the display in the *Current Cursor Position* field and *Current Field Length* field. *Current Cursor Position* shows the position of the cursor relative to the report, not the window. These fields enable you to determine where you are on the report.

A display area to the left of the report window lists the names of all of the print lines you have defined for this report. These are sorted into an appropriate sequence from the top to the bottom of the report page. If you defined more than 12 print lines, they will not all be displayed in the window.

When you first start the report painter, the window displays the upper left corner of the report.

## Using the Window Keys

Since the upper region of the screen is a window into the report, you need to scroll the window left or right to see more of the report. For many detailed reports you also need to scroll the window up and down to see all of the lines defined for the report.

The **Window Keys** function keys allow you to scroll the report window. These keys allow you to define the distance and direction in which to scroll the window.

When you press the **Window Keys** function key you will see the following keys displayed. Keys **Up Window** to **Right Window** are self explanatory.

```
Report Painter -- report product_rept [Application training]
Up      Down  Left  Right  4* 10  Centre Set Horz Set Vert  Main
Window Window Window Window Window Scroll  Scroll  Keys
```

### Report Painter Window Keys

Pressing **Centre Window** locates the center of the report in the window. This is the horizontal and vertical center.

Pressing **Set Horz Scroll** prompts you for the number of columns to skip whenever you press the **Left Window** or **Right Window** function keys. The default is 10.

Pressing **Set Vert Scroll** prompts you for the number of lines to skip whenever you press the **Up Window** or **Down Window** function keys. The default is 2.

---

## Task 3 - Painting the Product Report

The following layout shows the report you are about to paint.

```

      1          2          3          4          5          6          7          8
P1.01:                               Product File Details           Page: NNNN
P1.02:                               Date: AAAAAAAAAA

C1.01:  Product  Description                               Supplier Lead Time

D1.01:  AAAAAA
D1.02:  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  NNNNNN  NN Days

TF.01:  Total Number of Products  NNNN
```

The following pages describe the procedure for creating this report layout.

### 6-20 HP ALLBASE/4GL Reports

## Step 1 - Entering Text Items

As with the screen painter, you create text items on a report by simply positioning the cursor at the beginning of the item, pressing **Return** and entering the required text. The painter automatically recognizes the entered text as a literal.

You must have sufficient space in the window to enter a text item. You cannot create a text item that is longer than the width of the window, but you can create two text items that span across two windows for long report headings.

The first text item you need to enter on this report extends from column 33 to column 52. To create this item, this range of columns must be visible in the current window. Enter the text items by following these steps.

### To enter the text items for header lines:

Action	Explanation
Press <b>cursor home</b> and then press <b>Return</b>	The top left section of the report is displayed in the window.
Press <b>Window Keys</b>	The window keys function keys are displayed.
Press <b>Right</b> <b>Window</b>	This moves the window 10 columns to the right. This displays the range of columns needed to input the next text item.
Press <b>cursor right</b>	Keep doing this until the cursor is located in report column 33. (The current cursor position field contains the number of the column the cursor is currently positioned at. This field is at the right hand side of the screen, just below the paint region. You must press <b>Return</b> to update this cursor position display field.

Action	Explanation
Press <b>(Return)</b>	The painter enters the <i>text input</i> mode. (The screen shows an underline where you can input text.)
Type <b>Product File</b> <b>Details</b>	
Press <b>(Return)</b>	This terminates the entry of the text item.
Press <b>(Shift)+(cursor home)</b> and then press <b>(Return)</b>	The bottom right section of the report is displayed in the window.
Press <b>(cursor up)</b>	Keep pressing this key until the cursor is located on line P1.01.
Press <b>(cursor left)</b>	Keep pressing this key until the cursor is located at column 65.
Press <b>(Return)</b>	This updates the cursor position display. The painter enters <i>text input</i> mode.
Enter <b>Page:</b>	
Move to P1.02 col 65	
Press <b>(Return)</b>	
Enter <b>Date:</b>	

## 6-22 HP ALLBASE/4GL Reports

This completes the entries for the text items that appear on the two header lines of the report.

Now you can enter the remaining text items for the C1.01, D1.02, and TF.01 lines.

**To enter the text items for the remainder of the report:**

1. Use the cursor keys to move the cursor about the window.
2. Use the window keys to move about the report when you hit the window edge.
3. Use the cursor positions listed below as a guide.

<b>Line Group</b>	<b>Text</b>	<b>Cursor Position</b>
C1.01	Product	003
	Description	013
	Supplier	045
	Lead Time	057
D1.02	Days	061
TF.01	Total Number of Products	003

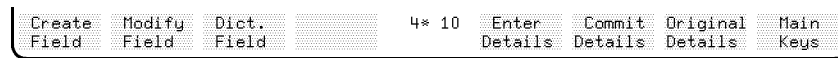
**Step 2 - Defining Data Fields**

Next, you need to create the data fields on the report. This can be done in a number of ways with the field keys.

**To create data fields:**

1. Position the cursor at line P1.01, column 71.
2. Return to the main function key set and press the **Field Keys** function key.

The following function keys are displayed:



### Field Keys

The **Create Field** and **Dict. Field** function keys allow you to define data fields on the report line. The function keys **Enter Details**, **Commit Details**, and **Original Details** allow you to define the data that is printed in the field and any formatting or totalling that is required for this field.

3. With the cursor still at line P1.01, column 71, complete the following steps.

Action	Explanation
Press <b>Create Field</b>	An inverse video area extends to the right of the cursor in the report painter window.
Type <b>NNNN</b>	This specifies a numeric field 4 digits long.
Press <b>Return</b>	This completes entry to the field. HP ALLBASE/4GL should display the message: <i>Item passed validation</i>

The character you use to define the field identifies the type of data that the field can contain. You can use the following characters:

Character	Field Type
A	Alphanumeric
N	Numeric
Z	Zero suppressed numeric

The message that you receive confirms that you have entered a valid field definition.

Now enter the details for this field using the **Enter Details** function key.

## 6-24 HP ALLBASE/4GL Reports

1. Position the cursor anywhere on the field (the current field will be highlighted) and press the **Enter Details** function key.

The painter moves the cursor to the *Field Name/Literal* field in the lower portion of the screen.

2. Now step through the fields and make the necessary entries.

**To enter the field details:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Field Name/Literal</b>	*PAGENO	Specifies the source of data for the report field. In this case, it is the communication area field *PAGENO. This field contains the number of the report page currently being printed.  You will notice that when the current report field is a literal, this field contains a message: <i>Literal - see line format</i> .
<b>Edit Code</b>	Accept the default.	This field has the same meaning as any other HP ALLBASE/4GL edit code field. The value in this field defaults to either N or X depending on the character you used to paint the field.
<b>Justification</b>	Accept the default.	This defaults to a standard justification code depending on the character used to paint the field.
<b>Pad Character</b>	0 (zero)	This character fills blank spaces in the field after it has been justified in the correct direction.

Field	Entry	Explanation
<b>Underline</b>	Leave blank.	This is similar in meaning to the same field on the report line header screen. However, the underlining or overlining defined here only applies to this report line field, not to all the fields on the line.
<b>Print Field Control</b>	Accept the default.	This entry defines when this field should be printed. <i>A</i> means always, or each time the line is printed. <i>C</i> means only when the value in the field changes.  This feature is useful in the detail lines (D1 - D9), since printing the information for a field only when its value changes makes the report appear less cluttered.
<b>Total Number</b>	Leave blank.	The automatic totalling facility for totalling fields down the report. The number you enter here relates to the communication area fields *TOTALS(1) to *TOTALS(16). When left blank it indicates that no totalling takes place for this field.
<b>Cross Add Number</b>	Leave blank.	The automatic totalling facility for use across the print line. The number you enter here relates to the communication area fields *CROSS(1) to *CROSS(5). When left blank it indicates that no totalling takes place for this field.

**To complete and commit this field:**

1. Having entered the details for this report line field, commit them by pressing the **Commit Details** function key.

The painter returns the cursor to the report line field. The details for the field remain displayed.

2. To display the details for a field, press **Return** while the cursor is on the field.

**6-26 HP ALLBASE/4GL Reports**



### In Summary

- To enter details for a field press the **Enter Details** function key while the cursor is on the appropriate report line field.
- To return to the painter mode, press the **Commit Details** function key to accept the values you have entered, or press the **Original Details** function key to abandon that entry of field details.
- The field's original details will be restored after you press the **Original Details** function key.

### To create the field that displays the print date:

Now you are ready to create the field that will display the print date for the report. Position the cursor at line P1.02, column 71 and perform the following steps:

Action	Explanation
Press <b>Create Field</b>	An inverse video area extends to the right of the cursor in the painter window.
Type <b>AAAAAAAA</b>	This specifies an alphanumeric field 8 characters long.
Press <b>(Return)</b>	The painter displays the message: <i>Item passed validation.</i>
Press <b>Enter Details</b>	This will move the cursor to the field details portion of the screen.
Enter <b>*DATE</b>	This is a reference to the communications area field that contains the current formatted date.
Enter <b>D</b> in the edit code field.	This formats the date according to the system wide date format.
Press <b>Commit Details</b>	No further details are needed for this field. Commit the details for this field and return the cursor to the painter window.

The page heading and column heading lines are now complete. You can now complete the total line.

1. Position the cursor at line TF.01, column 32 and create a four digit numeric field that refers to the communication area field \*COUNT(1).

This field maintains a count of the number of times the D1.01 line is printed.

The fields for the D1 lines will be created by referring to the dictionary field specifications.

### Creating Fields Using Dictionary Field Specifications

You can create a data field on a report using a dictionary field specification name. The painter creates a field of the appropriate length and type. The painter also automatically constructs a data source field for the report line field. If the source field is not correct, you can easily correct it.

This is the prompt you will see when entering the field specification name:

```

Enter dictionary name
product_no
Create  Modify  Dict.  24* 11  Enter  Commit  Original  Main
Field  Field  Field  Details  Details  Details  Keys
  
```

#### Dictionary Specifications Window

#### To enter fields using dictionary specifications window:

1. Position the cursor at line D1.01, column 3. You will have to display the window function keys, and use them, to move to this position.
2. Then perform the following steps from the **Field Keys** function key set:

Action	Explanation
Press <b>Dict Field</b>	You are prompted to enter the name of a dictionary field specification.
Type <b>product_no</b>	This is the dictionary field specification name for this report field.
Press <b>(Return)</b>	The report painter automatically creates a field of the appropriate length and type on the screen. It creates a data source in the <i>Field Name/Literal</i> field.

This field source defaults to the form *F-field name.primary file name*. In this case the source created will be *F-product\_no.product*.

This is exactly what you require, so the report line field is complete.

**To complete the remaining lines:**

1. Position the cursor at line D1.02, column 13 and follow the same procedure to create a field using the field specification name *description*.
2. Position the cursor at line D1.02, column 46 and create a field using the field specification name *supplier\_no*.
3. Now create the last field on this report. At line D1.02, column 58, create a field using the field specification name *lead\_time*.

This completes the painting of this report.

**Step 3 - Saving the Report**

To save this report image:

1. Return to the main set of function keys by pressing the **Main Keys** function key.
2. Then press the **Exit** function key.

This automatically saves the report under the name that it was called up with. You are then returned to the Reports Menu. Now generate this report as explained in the next lesson.

---

## Summary

In this lesson you used the report painter to create the image of a report.

The report painter lets you position report fields and literals on the lines of the report. The painter also lets you specify the following details of fields on report lines:

- The source of data to be printed in the field.
- The edit code and justification for the field.
- The automatic totalling performed for the field.

To create a literal on a report line, position the cursor at the starting point of the literal, press **Return** and enter the text for the literal. To create output fields on a report line, you can define the field directly, or use a dictionary field specification to define the field.

---

## Lesson 13 - Generating and Testing a Report

---

### Objectives

When you have completed this lesson, you will have learned how to:

- Generate a report.
- Test the report using the application testing mode.
- Use the menu bypass system as a quick way of testing application items in the application testing mode.

---

### Generating a Report

A report is another item that requires generation. You cannot generate a report before you have defined all the items for the report.

#### To generate a report:

1. Activate the *Header* item from the reports menu.

The report header screen has a function key that allows you to generate the report after you have defined all of its constituent parts. You will also find this function key on some of the other report screens.

2. On the report header screen, enter the name of the report to be generated, if not already displayed, then press the **Generate Report** function key.

As with the generation of other items, HP ALLBASE/4GL displays a message verifying that the report is being generated.

If you have specified everything correctly, the report will be generated successfully. However, if there are any errors, HP ALLBASE/4GL displays an error screen.

The error details displayed will specify which part of the report is in error (for example header, sorting, links selection, line header, or the line details).

A listing below the error details shows the element that was found to be in error, and the actual error message.

3. If you encounter an error, note it then go to the appropriate screen and correct the problem.
4. When you have corrected it, generate the report again.

When the report has been successfully generated you can test it, using the instructions below.

---

## Testing a Report

There are two possible ways to test this report. The quickest method is to go into *Test* mode and use the developer menu bypass facility to execute any item from a menu.

A second way is to create a menu item on the *main* menu of your application to run the report. Both methods are described here. Try both methods.

### Testing by Menu Bypass

This is the quick way available to you as a developer. This method is not available to end users.

From the developer main menu, follow the steps below to test the report:

Action	Explanation
Activate <i>Test</i>	This starts up your application in developer testing mode.
Press <b>F7</b> and then press <b>Return</b>	This is the developer menu bypass facility command. You are prompted for the action you wish to execute.
Type <b>R-product_rept</b>	This refers to the report <i>R- product_rept</i> that have just created.
Press <b>Return</b>	This executes the <i>product_rept</i> report.

## 6-32 HP ALLBASE/4GL Reports

When the report commences, HP ALLBASE/4GL displays a message stating that the report is being prepared. The report is then displayed on the screen. When the report is finished, the menu screen is redisplayed.

## Testing from a Menu

This is an alternative way of running the testing procedure. It involves creating a new menu item on the main application menu to execute the report.

To do this, start up the screen painter and follow these steps:

Action	Explanation
Enter <b>main</b>	This is the name of the screen that you want to modify. The painter displays the screen in its current state.
Press <b>Main Keys</b>	This displays the main painter keys.
Move cursor to line 9, column 32	This provides adequate spacing away from the prompt that is already there.
Press <b>Action</b>	This displays the action item prompt at the base of the screen.
Enter <b>R</b>	This retrieves the report action and then moves the cursor to the next field.
Enter <b>product_rept</b>	This creates an action item that executes the <i>product_rept</i> report. The prompt is the short description of the report from the report header screen.
Press <b>Return</b>	This terminates the creation of the action item prompt.
Enter <b>Product Report</b>	This creates the text for the action item prompt.
Press <b>Exit</b>	This saves the screen and exits from the painter.

Having created this menu item, activate the *Test* option on the developer main menu and run the report by activating the report option on the application menu.

When the report commences, HP ALLBASE/4GL displays a message stating that the report is being prepared. The report is then displayed on the screen. When the report is finished, the application *main* menu is redisplayed.

---

## Summary

In this lesson you generated and tested a report.

During generation, HP ALLBASE/4GL resolves references to data items in the report and also produces an executable form of the report.

After you generated the report, you tested it.

This lesson introduced the menu bypass facility in the application testing mode. The menu bypass facility allows you to execute an item directly. You can do this by pressing **[ ]** and then **[Return]** at any menu. HP ALLBASE/4GL displays a message asking you to enter an action prefix, and an action name. The menu bypass system allows you to execute any action that can be called from a menu.

## From Here On

This completes the development of the *training* application. To check your understanding of the material in this chapter, turn to the Self Test Questions on the next page.

The lessons in the remaining chapters show you additional capabilities of the HP ALLBASE/4GL application. You can begin them now, or return to them after practicing what you have learned so far.

If you need additional information about developing an application, refer to your *Developer Reference Manual*. This two-volume manual provides detailed explanation of all facets of HP ALLBASE/4GL.



---

## Self Test Questions

**Question 1.** There are six facilities that you can use to define an HP ALLBASE/4GL report. List the facilities, and identify which ones you must always use to create a report?

**Question 2.** Is the following sentence true or false?

*C* report lines cannot contain numeric constants or variables.

**Question 3.** What are two methods of creating a data field in a report?

**Question 4.** Which must you do first; define the report line headers, or paint the report.

**Question 5.** What functions do the \*BYPASS switch and the \*ENDLINE switch perform?

**Question 6.** Which of the following can you specify on the line header screen:

- Suppress line if all values are blank.
- Print field control.
- Pad character.
- Blank lines before line.

**Question 7.** How do you initiate the developer menu bypass facility command?

**Question 8.** How many detail line groups can a single report contain? How many physical lines may each detail line group contain?

---

## Answers

**Answer 1.** The following six facilities can be used to define a report:

- Report header.
- Report line header.
- Report painter.
- Report sorting.
- Report selection.
- File linkages.

The first three of these are necessary for all reports.

**Answer 2.** True. HP ALLBASE/4GL treats type *C* lines as literals for their entire length.

**Answer 3.** A data field can be created by:

- Pressing the **Create Field** function key and entering *A* or *N* characters.
- Pressing the **Dict. Field** function key and entering a dictionary field specification name.

**Answer 4.** The report line headers must be defined before the report can be painted.

**Answer 5.** \*BYPASS controls the printing of each print line and bypasses the current print line when set *on*.

\*ENDLINE controls the line group, and terminates the processing of the line group when set *on*.

**Answer 6.** Suppress line if all values are blank, and blank lines before line, can be specified on the line header screen. The other options can be specified on the screen painter screen.

**Answer 7.** To initiate the menu bypass facility, press **[/]** and then press **[Return]**.

**Answer 8.** Each report can contain nine detail line groups. Each of these detail line groups can contain up to 99 physical lines.

### 6-36 HP ALLBASE/4GL Reports

## Learning More Features

---

In Chapters 1 through 6, you learned how to develop a basic application that uses a process, a screen, a function, and a report. Together, these items allow you to enter data into the *product* file, and to produce a simple report of the data contained in the file.

The four lessons in this chapter will show you how to modify this application so that it offers more functionality and user feedback.

This chapter will show you how to use the appropriate HP ALLBASE/4GL screens to add the following items to the *training* application:

- Storage items such as:
  - Variables.
  - Constants.
  - Application titles.
- Messages.
- Validation ranges.

To use these features, you will learn how to expand the logic blocks that you have already written and how to write additional logic blocks. You also will learn how to modify one of the existing field specifications.

Working with these features will give you additional practice in using HP ALLBASE/4GL and help you learn how to use more complex procedures for developing applications. If you do not understand a concept presented in these chapters, check your *Developer Reference Manual*.

---

## How Will the Modifications Look?

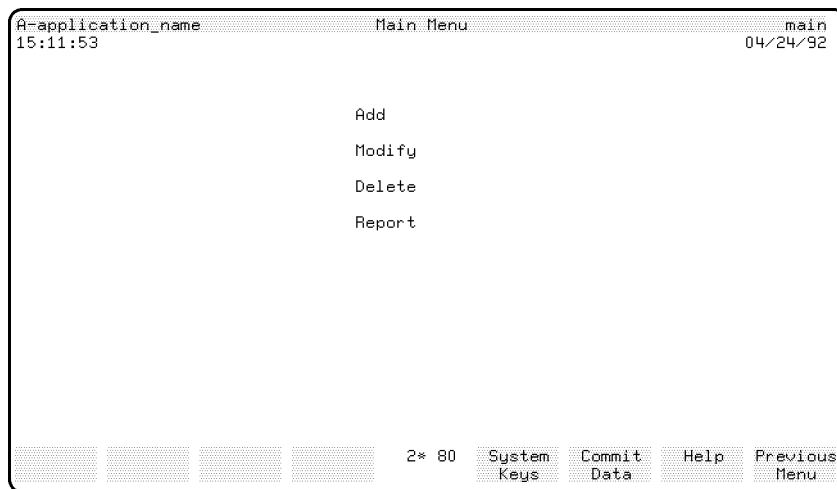
Currently, the *product\_scrn* screen is very simply defined and driven. It doesn't allow a user to delete a record from the product file, and it doesn't provide much operational information for the user. If the application is HP ALLBASE/SQL based, it doesn't allow a user to update an SQL record.

Apart from the data displayed on the screen, there is very little to indicate that a record is being added or modified. In addition there is no provision for trapping any file access errors that may occur.

The enhancements that you define in this chapter allow the user to specify that a record is to be added, modified, deleted or reviewed.

The main menu will offer five choices: *Add*, *Modify*, *Delete*, *Review*, and *Report*. This allows the system administrator to assign menu item security to each menu item, limiting access to selected items for particular users. The menu item security system is described in the *HP ALLBASE/4GL Developer Administration Manual*.

The main menu is shown here.



**Modified Main Menu for Training Application**

## 7-2 Learning More Features

This menu allows the user to select the following options.

<b>Operation</b>	<b>Description</b>
Add	Add a product.
Modify	Modify an existing product.
Delete	Delete a product.
Review	Review a product.
Report	Report on a product.

### **The Application Screens**

When the user selects an option from the main menu, the appropriate update mode is set. The *product\_scrn* data screen is then displayed for each mode.

```
A-application_name      Product Details      product_scrn
10:24:59                04/27/92

      Product Number  .....

      Description      .....
      Supplier Number .....
      Lead Time        .....

..... 24* 75  System  Commit  Help  Previous
          Keys   Data   Menu
```

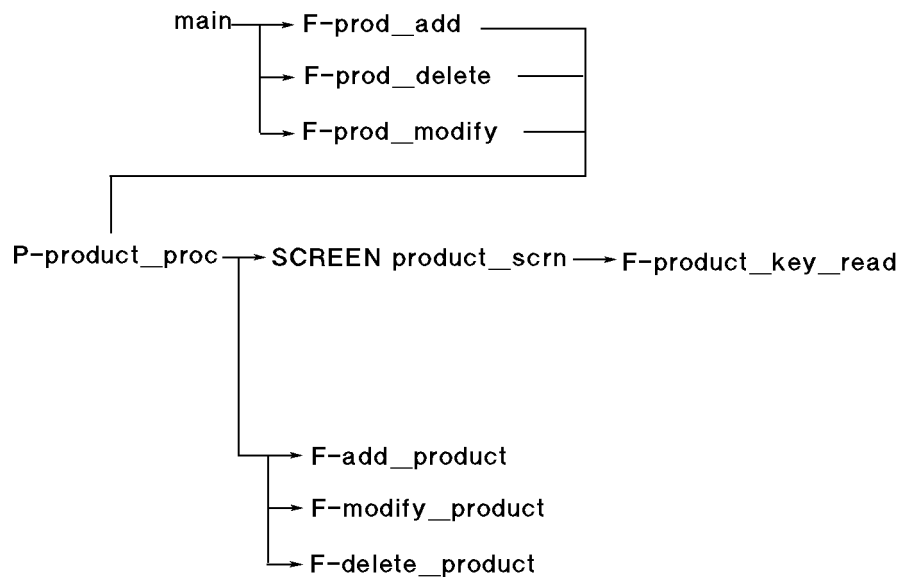
**product\_screen Screen**

In addition, when records are added, modified, deleted, or reviewed the user will receive messages that indicate whether the task was successful or not.

### **Learning More Features 7-3**

## Application Structure

The following diagram shows the structure of the extensions to the application.



Each menu item calls a function. These functions set the appropriate mode by initializing a variable. Each function then calls the *product\_proc* process.

The process displays the *product\_scrn* screen.

The first lesson introduces you to additional dictionary items that you will need to use in the new functions you will create in lesson 14.

## 7-4 Learning More Features

---

## Lesson 13 - Creating Storage Items

---

### Objectives

When you have completed this lesson, you will have learned how to define the following storage items.

- An application title.
- Two variables.
- One numeric constant.
- Six alphanumeric constants.

These items will be used in the *product\_proc* logic block that you will modify later, and in some new logic blocks that you will create.

---

### What Are Storage Items?

The dictionary storage items are somewhat different from the other dictionary entries. In contrast to dictionary field specifications, storage items do hold application data. The dictionary storage items are:

- Variables.
- Calculated items.
- Numeric constants.
- Alphanumeric constants.
- Scratch-pad field names.
- Application titles.
- Work areas.

HP ALLBASE/4GL variables are similar to variables in other programming systems. You can specify the number of characters, the edit code, and number of decimal places for a variable.

Calculated items are variables that are evaluated each time they are called by an application. The value of a calculated item is the result of an arithmetic expression or a logic function that is evaluated when the item is called.

The scratch-pad contains up to 99 scratch-pad fields. Each scratch-pad field is a temporary storage area that you can use for storing data within an application.

Each scratch-pad field takes on the attributes of the data written into it, and the combined scratch-pad only uses the amount of system memory required to hold the data contained within it at any given time.

---

## Defining Application Titles

An application title is a literal string that can be displayed on any screen within an application or printed on reports. Any change made to an existing application title is immediately reflected through the whole application and related versions that use the application title. You can only reference an application title directly with the screen painter.

### Menu Path

#### To access the Application Titles Screen:

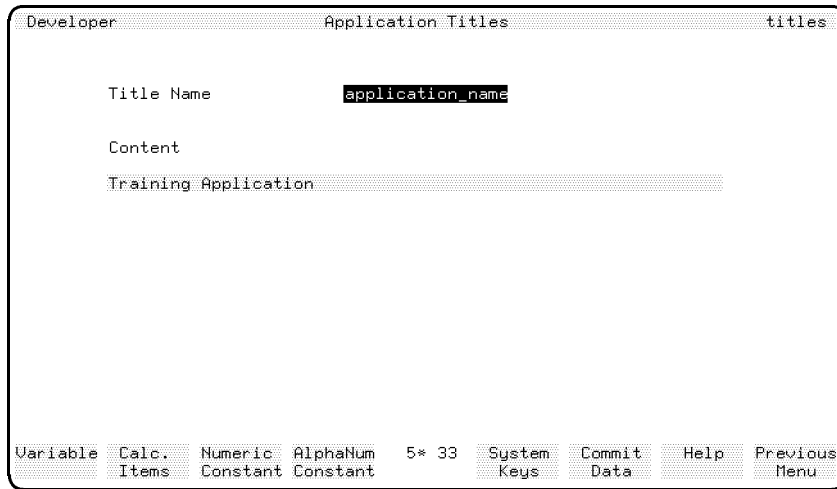
1. From the main menu, select the *Dictionary* option.
2. Select the *Storage Items* option.
3. Choose *Titles*.

### Screen Description

An application title is defined by its name and contents. The contents are stored as an alphanumeric string of length equal to the number of characters specified in the string.

## 7-6 Learning More Features





### Application Titles Screen

To enter the field values:

Field	Entry	Explanation
Title Name	application_name	The name of the application title that is used by the application.
Content	Training Application	The actual application title. Any trailing spaces are stripped when you commit the title.

To create the application title and proceed:

1. Press the **Commit Data** function key.

When you next run your application you will see that the screens contain the new definition of this item.

2. Press the **Previous Menu** function key to return to the storage items menu.

---

## Defining Variables

You can now create a variable called *product\_status*. This variable is set after a file access to indicate if a record exists or not.

### Menu Path

**To access this screen from the main menu:**

1. Select the *Dictionary* option.
2. Select the *Storage Items* option.
3. Choose *Variables*.

### Description

Variables have a fixed length and type. When an application starts, alphanumeric variables are initialized to spaces and numeric variables to zero. Variables are not initialized at any other time.

```
Developer Variables variables
Variable Name      product_status
Length            1
Edit Code         X (X/A/U/K/N/S/Q/D/T)
Number of Decimal Places
Pad Character     █
Work Areas  Calc. Items  Numeric Constant  AlphaNum Constant  13* 48  System Keys  Commit Data  Help  Previous Menu
```

**Variables Screen**

## 7-8 Learning More Features

**To enter the field values:**

Field	Entry	Explanation
Variable Name	product_status	When referred to in the application, a variable is always prefixed with V-. The full reference to this variable is V-product_status.
Length	1	The length of a variable is fixed, and the maximum possible length is 99 characters.
Edit Code	X	This edit code has the same definition as all other edit codes.
Number of Decimal Places	Leave blank.	Variables can use up to nine decimal places if they are numeric.
Pad Character	Leave blank.	HP ALLBASE/4GL automatically justifies a variable when it receives a value. Alphanumeric variables are left justified, and numeric variables are right justified.  This field allows you to specify a pad character which is used to fill the remaining spaces in the variable after it has been justified. The pad character must be a valid character for the edit code specified.

**To create the variable:**

1. Press the **Commit Data** function key.

**Defining the Mode Variable**

Next, you will create a second variable called *mode*, which will be used to indicate the current update mode (one of Add, Modify, Delete, or Review).

**To define the mode variable:**

1. Enter the values for fields shown below that require an entry.

2. For all other fields, simply press the **Return** or **Tab** key to accept the default.

<b>Field</b>	<b>Entry</b>
Name	<b>mode</b>
Length	<b>6</b>
Edit Code	<b>X</b>

**To create this variable and continue:**

1. Press the **Commit Data** function key.
2. Then press the **Previous Menu** function key to return to the storage items menu.

---

## Defining Numeric Constants

### Menu Path

**To access the Numeric Constants Screen:**

1. From the main menu, select the *Dictionary* option.
2. Select the *Storage Items* option.
3. Choose *Numeric Constants*.

### Description

A numeric constant is defined as a signed or unsigned number of fixed value with a fixed number of decimal places. A numeric constant can be used to initialize other variables and fields, or it can be used as a fixed comparative value when you want to test another field.

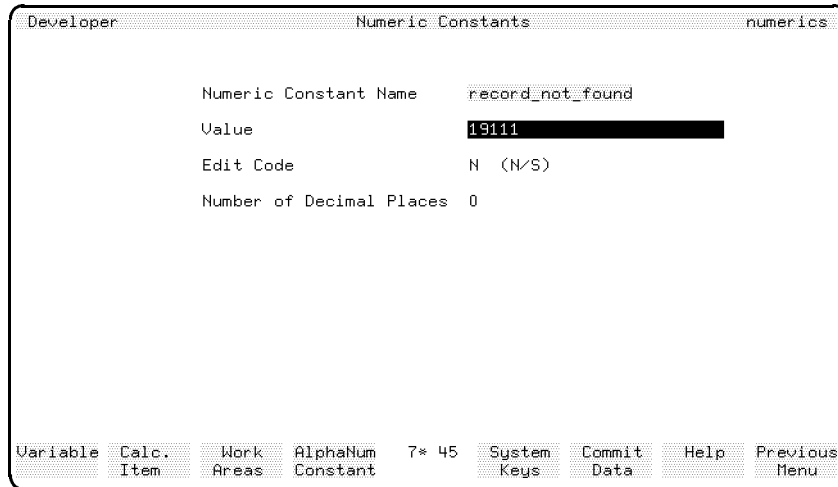
The value of the numeric constants you create will be different for each data manager.

If you are creating the KSAM based application or the HP TurboIMAGE/iX based application, read on below to create the *record\_not\_found* constant.

## 7-10 Learning More Features

If you are creating the HP ALLBASE/SQL based application, create the *end\_of\_file* variable in “HP ALLBASE/SQL Based Applications”.

## KSAM Based and HP TurboIMAGE/iX Based Applications



**Numeric Constants Screen**

To enter the field values:

Field	Entry	Explanation
<b>Numeric Constant Name</b>	record_not_found	When referred to in the application, a numeric constant name is prefixed with <i>N-</i> . The full reference to this constant is <i>N- record_not_found</i> .
<b>Value</b>	19111	The value of this numeric constant is the file error value returned by the data manager when a requested record is not found. In this application, the numeric constant is used to test the file return status value.

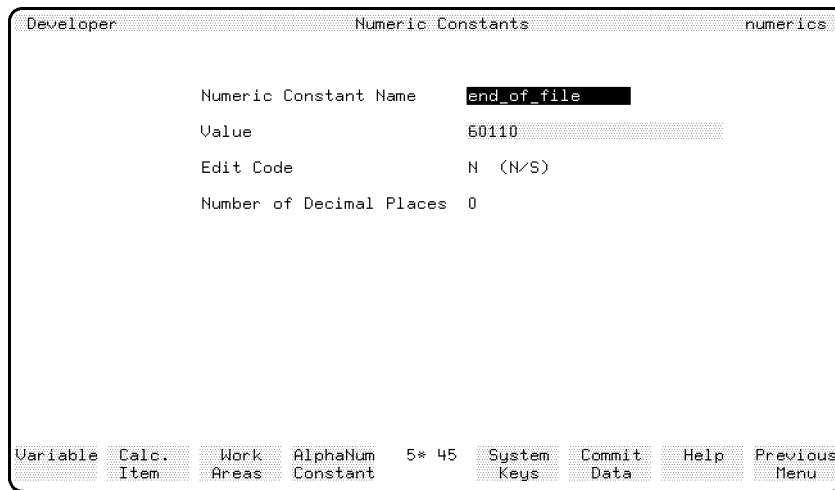
**Learning More Features 7-11**

Field	Entry	Explanation
<b>Edit Code</b>	Accept the default.	This is a display only field that shows the edit code of the variable. If the variable value includes a + or a - sign, the display in this field is <i>S</i> .
<b>Number of Decimal Places</b>	Accept the default.	Display only field that shows the number of decimal places in the variable. Numeric variables can have up to 9 decimal places.

**To create the numeric constant and continue:**

1. Press the **Commit Data** function key.
2. Press the **Previous Menu** function key to return to the storage item menu.
3. Then turn to the section called “Defining Alphanumeric Constants”.

**HP ALLBASE/SQL Based Applications**



**Numeric Constants Screen**

**7-12 Learning More Features**

**To enter the field values for SQL:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Numeric Constant Name</b>	<code>end_of_file</code>	When referred to in the application, a numeric constant name is prefixed with <i>N-</i> . The full reference to this constant is <i>N- end_of_file</i> .
<b>Value</b>	60110	The value 60110 is the data manager error value returned if a FILE *NEXT command encounters the end or beginning of a file for the active set. In this application, the numeric constant is used to test the file return status value.
<b>Edit Code</b>	Accept the default.	Display only field that shows the edit code of the variable. If the variable value includes a + or a - sign, the display in this field is <i>S</i> .
<b>Number of Decimal Places</b>	Accept the default.	Display only field that shows the number of decimal places in the variable. Numeric variables can have up to 9 decimal places.

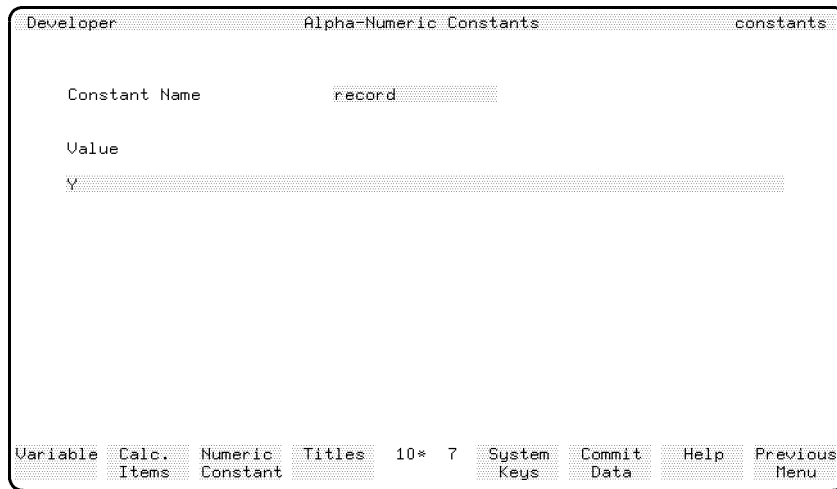
**To create numeric constant and continue:**

1. Press the **Commit Data** function key.
2. Press the **Previous Menu** function key to return to the storage item menu.

---

## Defining Alphanumeric Constants

Alphanumeric constants are used in the same way as numeric constants, but they contain alphanumeric data.



**Alpha-Numeric Constants Screen**

### Menu Path

**To access the screen from the main menu:**

1. Select the *Dictionary* option.
2. Select the *Storage Items* option.
3. Choose Alphanumeric Constants.

### Description

An alphanumeric constant is defined in the same way as an application title. The contents are stored as an alphanumeric string of length equal to the number of characters entered in the string. Any trailing spaces are stripped away when you commit the constant.

## 7-14 Learning More Features



**To enter the field values:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Constant Name</b>	record	When referred to in an application, an alphanumeric constant is prefixed with <i>C-</i> . The full reference to this constant is <i>C- record</i> .
<b>Value</b>	Y	The entire content of this constant. It is a flag value indicating that a record has been found. You will see how it's used later.

**To complete the alphanumeric constant:**

1. Press the **Commit Data** function key.

### **Creating Additional Alphanumeric Constants**

**To create the following alphanumeric constants:**

1. Enter the information listed for a constant.
2. Press the **Commit Data** function key for that constant.
3. Continue this process for each alphanumeric constant listed below.
  - a. *no\_record* containing the value **N**.
  - b. *add* containing the value **Add**.
  - c. *modify* containing the value **Modify**.
  - d. *delete* containing the value **Delete**.
4. Press the **Previous Menu** function key to return to the storage definition menu.

Now you have completed the definition of the required storage items for the enhanced application.

---

## Summary

In this lesson you defined the following storage items:

- An application title.
  - *application\_name*.
- Two variables.
  - *product\_status*,
  - *mode*.
- One numeric constant.
  - *end\_of\_file* (for the HP ALLBASE/SQL based application),
  - *record\_not\_found* (for the KSAM and HP TurboIMAGE/iX based applications).
- Five alphanumeric constants.
  - *record*,
  - *no\_record*,
  - *add*,
  - *delete*,
  - *modify*.

All storage items are defined using the storage items menu in the dictionary.

### Application Titles

Application titles are literal strings that can be used on screens and reports. Any change to an application title is immediately reflected on screens and reports that use the title.

### Variables

All variables have a fixed length and type. The type of data a variable can contain is determined by its edit code.

HP ALLBASE/4GL initializes numeric variables to zero, and alphanumeric variables to spaces, when the application starts. They are not initialized at any other time.

## 7-16 Learning More Features

## **Constants**

You can define numeric constants and alphanumeric constants. Numeric constants can contain numeric values and the characters - and + .

Alphanumeric constants are literal strings, and can contain any printable characters.

---

## Lesson 14 - Modifying a Logic Block

---

### Objectives

When you have completed this lesson you will have learned how to use the process details screen to expand and modify the steps in the *product\_proc* process.

---

### Modifying the *product\_proc* Process

The *product\_proc* process needs to be modified from the point of exit from the *product\_scrn* screen. When the user terminates the screen, the process will check which mode the user has selected, and then update the file accordingly. After the file update, the process will display a message to confirm its success. If an error occurs during the update, the process will display an appropriate message.

If you are developing the KSAM based application, continue reading below.

If you are developing the HP ALLBASE/SQL based application, refer to the “HP ALLBASE/SQL Based Applications” section for a description of the changes to the process.

If you are developing the HP TurboIMAGE/iX based application, refer to the “HP TurboIMAGE/iX Based Applications” section for a description of the changes to the process.

### KSAM Based Applications

The modified *product\_proc* process reads like this:

```
1 MODE *WRITE product
2 MOVE C-no_record V-product_status
3 SCREEN product_scrn
```

### 7-18 Learning More Features

```

4 IF V-mode = C-add THEN VISIT add_product
5 IF V-mode = C-modify THEN VISIT modify_product
6 IF V-mode = C-delete THEN VISIT delete_product
7 IF *IOSTATUS = "00000" | *IOSTATUS = N-record_not_found
    THEN ENTER 3
8 MESSAGE file_error
9 SCREEN main

```

The following paragraphs explain what each step in the process does.

- Line 1            This step specifies that the process can write to the *product* file.
- Line 2            This step initializes the value of the variable *V-product\_status* when the screen is first displayed.
- Line 3            This step displays the *product\_scrn* data screen. The same screen is used for each mode. The setting of the *mode* variable determines how the process updates the files when the user commits the screen.
- Lines 4, 5 and 6    The **IF** commands test the update mode, and then invoke the appropriate file update function using the **VISIT** command.
- Line 7            If the data manager returns either 00000 or the value of the constant, this step returns control to step 3 of the process to display the *product\_scrn* screen again.
- Values of 00000 or 19111 in \*IOSTATUS indicate that all file transactions occurred without error, or that the record could not be found.
- If the file manager return status is not one of these values, a file error has occurred during the update functions. In this case, control passes to the error handling logic starting at step 8.
- Line 8            Control only passes to this step if an error is detected at step 7. This step displays a message to warn the user that an error has occurred and the current product information may be corrupted.

Line 9            This step returns the user to the *main* menu. Displaying a menu is one technique you can use to recover from serious, but non-fatal error conditions. When HP ALLBASE/4GL displays a menu, all current activity in the application ceases. This means that the application returns to a known condition. To continue from the *main* menu, the user must select a menu action. Any actions that can update data files initiate a process, so all file record and screen buffers are cleared automatically.

The following steps show you how to modify the existing process.

### **Modifying a Process**

This part of the lesson shows you how to use facilities that allow you to modify logic blocks. You use the same methods to modify processes and functions.

#### **To insert a step:**

Call up the process details screen and follow these next steps.

<b>Action</b>	<b>Explanation</b>
Enter <code>product_proc</code>	This displays the <i>product_proc</i> process in its current state and places the cursor in the step number field.
Enter <code>2</code>	The cursor moves to the action field.
Enter <code>I</code>	This indicates that you want to insert a step before step number 2. The current steps move down one line, and step 2 becomes blank. Then the cursor moves to the command field.
Enter <code>MOVE</code>	This displays an open window. The <code>MOVE</code> command is used to copy the contents of a field to another field.
Enter <code>C-no_record</code> <code>V-product_status</code>	

## **7-20 Learning More Features**

**To complete the window:**

The MOVE command is now complete.

1. Press the **Commit Data** function key to commit it.

HP ALLBASE/4GL clears the window and displays the new command at step 2. The cursor returns to the *Step Number* field and the step number is incremented to 3.

**Step Number References**

Read the last line of the process. Before you inserted a new step, this logic step was ENTER 2. Now it is ENTER 3.

HP ALLBASE/4GL has automatically incremented the step number reference as the SCREEN command has moved down from step 2 to step 3. HP ALLBASE/4GL updates step number references when you insert or delete lines from a logic block.

**Using the IF Window**

The next command that you need to insert is step 4. This contains an IF command.

1. Insert a new step 4
2. Enter **IF** in the *Command* field.

This displays the IF command window.

3. Complete the fields in this window as explained in the following table.

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Condition 1</b>		The first three fields of the IF window establish the first condition that the IF command tests. In many cases it is the only condition.
<b>Data Name 1</b>	<b>V-mode</b>	The name of the object to be tested. In this case, the command tests the variable <i>mode</i> , which contains one of the four update modes; add, modify, delete or review.
<b>IF Test</b>	<b>=</b>	The test to be performed. In this case the test is whether or not the content of the <i>mode</i> variable equals the value in the <i>Data Name 2</i> field.
<b>Data Name 2</b>	<b>C-add</b>	Where an IF Test compares two operands with a relational operator such as "=", "<", ">", or "<>", you must enter the name of the second operand in this field.  In this case, the <i>mode</i> variable will be set to the contents of the <i>add</i> alphanumeric constant if the user selected <i>Add</i> on the main menu. The IF command tests to see whether this is the current value of V-mode.
<b>AND/OR</b>	Leave this field blank.	The IF command can test two conditions and combine the results using a logical AND or a logical OR connective. The symbols <b>&amp;</b> and <b> </b> represent the AND and OR connectives respectively. When you leave this field blank, the cursor skips the next three fields.

## 7-22 Learning More Features



Field	Entry	Explanation
<b>THEN Statement</b>	VISIT add_product	<p>When you leave the AND/OR field blank and press <b>(Return)</b>, a <i>THEN</i> prompt appears. The cursor is placed in the field to the right of this prompt. This is where you enter the commands to be performed when the test is resolved as true.</p> <p>You can execute any command at this stage except IFLOOP, SELECT, or another IF statement.</p> <p>You can specify multiple commands by separating the commands with a semicolon. For example, you could enter a string of commands in this field like this:</p> <p><b>MESSAGE</b> added_ok ; <b>ENTER</b> 10</p>
<b>ELSE Statement</b>	Leave blank.	<p>After you enter the THEN statement, the window displays an <i>ELSE</i> prompt and positions the cursor in the field beside the prompt. This is where you enter the commands to be performed if either test is resolved as false. This entry follows the same format as the THEN statement. However, it is not a required field. If you don't enter any ELSE commands, control passes to the next step in the process or function if the test is resolved as false.</p> <p>You will use this field in a later part of the training guide.</p>

**To complete the window and continue:**

The IF command is now complete.

1. Press the **Commit Data** function key to commit it. HP ALLBASE/4GL clears the window and displays the new command at step 4.

The cursor returns to the *Step Number* field and the step number is incremented to 5.

2. Now insert steps 5 and 6 shown below.

The two steps are very similar to the step you just inserted. They also call up functions to update the *product* file.

```
5 IF V-mode = C-modify THEN VISIT modify_product
6 IF V-mode = C-delete THEN VISIT delete_product
```

At the moment, there are only three update functions available, and the code is short and simple. However, in later lessons, you will introduce more functionality.

Instead of adding a lot of logic commands, you will convert these decision logic lines into one logic command that calls a decision table. Decision tables allow you to define a series of actions to be performed as the result of the outcome of a complex series of conditional tests or questions.

Your process should now look like this:

```
1 MODE *WRITE product
2 MOVE C-no_record V-product_status
3 SCREEN product_scrn
4 IF V-mode = C-add THEN VISIT add_product
5 IF V-mode = C-modify THEN VISIT modify_product
6 IF V-mode = C-delete THEN VISIT delete_product
7 FILE *WRITE product
8 ENTER 3
```

### **Continuing the Change**

The next task is to insert a new step 7. This step also uses the IF command, but this time, two sets of conditions are tested.

#### **To insert step 7:**

Use the following details for each field.

## **7-24 Learning More Features**

Field	Entry	Explanation
<b>Command</b>	IF	
<b>Data Name 1</b>	*IOSTATUS	In this case, the command tests the HP ALLBASE/4GL communication field *IOSTATUS, which contains either zero, or the error message number returned by the HP ALLBASE/4GL data manager after an access to an application data file.
<b>IF Test</b>	=	
<b>Data Name 2</b>	"00000"	The value "00000" will be the content of *IOSTATUS if no file errors have occurred. The IF command tests to see whether this is the current value of *IOSTATUS.
<b>AND/OR</b>		This represents OR.
<b>Condition 2</b>	Header Field	The next three fields establish a second condition that the IF command tests.
<b>Data Name 1</b>	*IOSTATUS	
<b>IF test field</b>	=	
<b>Data Name 2</b>	N- record_not_found	
<b>THEN Statement</b>	ENTER 3	When you complete the condition fields and press <b>(Return)</b> , a <i>THEN</i> prompt appears. Enter the command shown.
<b>ELSE Statement</b>	Leave blank.	

**To complete the window and continue:**

The IF command is now complete.

1. Press the **Commit Data** function key to commit it.

## Deleting a Step

You can now delete an existing step. We wish to delete the FILE step.

1. Make sure the cursor is at the *Step Number* field, and follow these next steps.

Action	Explanation
Enter 8	The cursor moves to the action field.
Enter D	This indicates that you wish to delete the current step.
Press <b>Return</b>	Step 8 is deleted from the logic block, and the steps below it are moved up.

A message is then displayed at the bottom of the screen, prompting you to press the **Commit Data** key to make the deletion permanent.

The D action blanks out the step from the screen. However, the step is not actually deleted from the logic block until you press **Commit Data**.

2. Press the **Commit Data** function key now.

## Modifying a Step

You can now change an existing step. The ENTER command at step 8 is no longer required. This step should now read as:

8 MESSAGE file\_error

There are two ways to alter this step. You could delete step 8 and then add another step 8 in its place. However, we shall use the modify facilities to alter this step.

## 7-26 Learning More Features

**To modify step 8:**

- Position the cursor at the *Step Number* field and follow the next steps.

<b>Action</b>	<b>Explanation</b>
Enter 8.	The cursor moves to the action field.
Press <b>Return</b>	Since the default action for an existing step is <i>Change</i> , the current contents of the step are displayed in an open window and the cursor is positioned in the first position of the window.
Press <b>Shift</b> + <b>Tab</b> , and then <b>Return</b>	This moves the cursor back to the command field.
Type <b>MESSAGE</b> over the top of the ENTER command	This alters the step to a MESSAGE command.
Press <b>Return</b>	This displays a new open window for the MESSAGE command.
Enter <b>file_error</b>	This is the message that will be displayed by this command.
Press <b>Commit Data</b>	Commit the step and HP ALLBASE/4GL will display the modified command.

## Adding Commands

Add the following command to the end of the logic block at step 9.

9 **SCREEN** main

This logic command uses an open window for you to enter the logic command details.

### To add the command:

1. Position the cursor on the *Action* field, regardless of step number, and display the whole process on the screen.
2. Enter the following information.

Action	Explanation
Enter L	The <i>L</i> action <i>lists</i> the current logic block to the screen. When you press <b>Return</b> , HP ALLBASE/4GL clears the screen and lists the entire process. Any lines that are too long to fit on the screen will be wrapped around to the next line.
Press <b>Return</b>	This returns you to the process details screen.

3. Press the **Generate Process** function key to generate the modified *product\_proc* process.

If you have entered the process exactly as it's listed at the beginning of this section, you won't receive any generate error messages. If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error.

This concludes this lesson. If you are using an SQL based application, continue with the next section. Otherwise, turn to the Summary section at the end of the lesson.

## 7-28 Learning More Features

---

## Modifying HP ALLBASE/SQL Based Applications

The modified *product\_proc* process will read like this:

```
1 MODE *WRITE product
2 MOVE C-no_record V-product_status
3 SCREEN product_scrn
4 IF V-mode = C-add THEN VISIT add_product
5 IF V-mode = C-modify THEN VISIT modify_product
6 IF V-mode = C-delete THEN VISIT delete_product
7 IF *IOSTATUS <> "00000" & *IOSTATUS <> N-end_of_file
    THEN ENTER 10
8 SQL commit
9 ENTER 3
10 MESSAGE file_error
11 SCREEN main
```

The following paragraphs explain what each step in the process does.

- |                  |   |
|------------------|---|
| Line 1           | This step specifies that the process can write to the <i>product</i> file.  |
| Line 2           | This step initializes the value of the variable <i>V-product_status</i> when the screen is first displayed.   |
| Line 3           | This step displays the <i>product_scrn</i> data screen. The same screen is used for each mode. The setting of the <i>mode</i> variable determines how the process updates the files when the user commits the screen.   |
| Lines 4, 5 and 6 | The IF commands test the update mode, and then invoke the appropriate file update function using the VISIT command.   |
| Line 7           | Values of 00000 or 60110 in *IOSTATUS indicate that all file transactions occurred without error, or that the end of the file was reached without finding the record.<br><br>If the file manager return status is neither of these values, a file error has occurred during the update functions. In this case, control passes to the error handling logic starting at step 10. |

**Learning More Features 7-29**

- Line 8            This step calls the SQL logic block *commit*, which issues a COMMIT WORK to make any table updates permanent in the database.
- Line 9            This step returns control to step 3 to display the *product\_scrn* screen again.
- Line 10           Control only passes to this step if an error is detected at step 7. This step displays a message to warn the user that an error has occurred and the current product information may be corrupted.
- Line 11           This step returns the user to the *main* menu. Displaying a menu is one technique you can use to recover from serious, but non-fatal error conditions. When HP ALLBASE/4GL displays a menu, all current activity in the application ceases. This means that the application returns to a known condition. To continue from the *main* menu, the user must select a menu action. Any actions that can update data files initiate a process, so all file record and screen buffers are cleared automatically.

The next steps show you how to modify the existing process.

### **Modifying a Process**

This part of the lesson shows you how to use facilities that allow you to modify logic blocks. You use the same methods to modify processes and functions.

#### **To insert a step:**

- Call up the process details screen and follow the next steps.

## **7-30 Learning More Features**



Action	Explanation
Enter <code>product_proc</code>	This displays the <i>product_proc</i> process in its current state and places the cursor in the step number field.
Enter 2	The cursor moves to the action field.
Enter I	This indicates that you want to insert a step before step number 2. The current steps move down one line, and step 2 becomes blank. Then the cursor moves to the command field.
Enter <code>MOVE</code>	This displays an open window. The <code>MOVE</code> command is used to copy the contents of a field to another field.
Enter <code>C-no_record</code> <code>V-product_status</code>	

#### To complete the window:

The `MOVE` command is now complete.

- Press the `Commit Data` function key to commit it.

HP ALLBASE/4GL clears the window and displays the new command at step 2. The cursor returns to the *Step Number* field and the step number is incremented to 3.

#### Step Number References

Read the last line of the process. Before you inserted a new step, this logic step was `ENTER 2`. Now it is `ENTER 3`.

HP ALLBASE/4GL has automatically incremented the step number reference as the `SCREEN` command has moved down from step 2 to step 3. HP ALLBASE/4GL updates step number references when you insert or delete lines from a logic block.

## Using the IF Window

The next command that you need to insert is step 4. This contains an IF command.

### To insert a new step 4:

1. Enter IF in the *Command* field.

This displays the IF command window. The fields on this window are explained in more detail below.

2. Complete the fields listed below.

Field	Entry	Explanation
Condition 1		The first three fields of the IF window establish the first condition that the IF command tests. In many cases it is the only condition.
Data Name 1	V-mode	The name of the object to be tested. In this case, the command tests the variable <i>mode</i> , which contains one of the four update modes; add, modify, delete or review.
IF Test	=	The test to be performed. In this case the test is whether or not the content of the <i>mode</i> variable equals the value in the <i>Data Name 2</i> field.
Data Name 2	C-add	Where an IF Test compares two operands with a relational operator such as "=", "<", ">", or "<>", you must enter the name of the second operand in this field.  In this case, the <i>mode</i> variable will be set to the contents of the <i>add</i> alphanumeric constant if the user selected <i>Add</i> on the main menu. The IF command tests to see whether this is the current value of V-mode.

## 7-32 Learning More Features

Field	Entry	Explanation
<b>AND/OR</b>	Leave this field blank.	The IF command can test two conditions and combine the results using a logical AND or a logical OR connective. The symbols <b>&amp;</b> and <b> </b> represent the AND and OR connectives respectively. When you leave this field blank, the cursor skips the next three fields.
<b>THEN Statement</b>	VISIT add_product	<p>When you leave the AND/OR field blank and press <b>(Return)</b>, a <i>THEN</i> prompt appears. The cursor is placed in the field to the right of this prompt. This is where you enter the commands to be performed when the test is resolved as true. You can execute any command at this stage except IFLOOP, SELECT, or another IF statement.</p> <p>You can specify multiple commands by separating the commands with a semicolon. For example, you could enter a string of commands in this field like this:</p> <pre>MESSAGE added_ok; ENTER 10</pre>
<b>ELSE Statement</b>	Leave blank.	<p>After you enter the THEN statement, the window displays an <i>ELSE</i> prompt and positions the cursor in the field beside the prompt. This is where you enter the commands to be performed if either test is resolved as false. This entry follows the same format as the THEN statement. However, it is not a required field. If you don't enter any ELSE commands, control passes to the next step in the process or function if the test is resolved as false.</p>

**To complete the window:**

The IF command is now complete.

1. Press the **Commit Data** function key to commit it.

HP ALLBASE/4GL clears the window and displays the new command at step 4. The cursor returns to the *Step Number* field and the step number is incremented to 5.

2. Now insert steps 5 and 6, shown below, which are very similar to the step you just inserted. These two steps also call up functions to update the *product* file.

```
5 IF V-mode = C-modify THEN VISIT modify_product
```

```
6 IF V-mode = C-delete THEN VISIT delete_product
```

At the moment, there are only three update functions available, and the code is short and simple. However, in later lessons, you will learn how to introduce more functionality.

Instead of adding a lot of logic commands, you will convert these decision logic lines into one logic command that calls a decision table. Decision tables allow you to define a series of actions to be performed as the result of the outcome of a complex series of conditional tests or questions.

Your process should now look like this:

```
1 MODE *WRITE product
```

```
2 MOVE C-no_record V-product_status
```

```
3 SCREEN product_scrn
```

```
4 IF V-mode = C-add THEN VISIT add_product
```

```
5 IF V-mode = C-modify THEN VISIT modify_product
```

```
6 IF V-mode = C-delete THEN VISIT delete_product
```

```
7 FILE *INSERT product
```

```
8 SQL commit
```

```
9 ENTER 3
```

### **Continuing the Change**

The next task is to insert a new step 7. This step also uses the IF command, but this time, two sets of conditions are tested.

## **7-34 Learning More Features**

**To insert step 7:**

<b>Field</b>	<b>Entry</b>	<b>Explanation</b>
<b>Command</b>	IF	
<b>Data Name 1</b>	*IOSTATUS	In this case, the command tests the HP ALLBASE/4GL communication field *IOSTATUS, which contains either zero, or the error message number returned by the HP ALLBASE/4GL data manager after an access to an application data file.
<b>IF Test</b>	<>	
<b>Data Name 2</b>	"00000"	The value "00000" will be the content of *IOSTATUS if no file errors have occurred. The IF command tests to see whether this is the current value of *IOSTATUS.

Field	Entry	Explanation
AND/OR	&	This represents AND.
Condition 2		The next three fields establish a second condition that the IF command tests.
Data Name 1	*IOSTATUS	
IF test	<>	
Data Name 2	N-end_of_file	
THEN Statement	ENTER 10	When you complete the condition fields and press <b>(Return)</b> , a <i>THEN</i> prompt appears. Enter the command shown.

#### To complete the window:

The IF command is now complete.

1. Press the **Commit Data** function key to commit it.

#### Adding Commands

The following commands can be added to the end of the logic block at steps 11 and 12 respectively. Both of these logic commands use open windows for you to enter the logic command details.

1. Add the commands as shown below.

11 MESSAGE file\_error

12 SCREEN main

#### Deleting a Step

You can now delete an existing step. We wish to delete the FILE step.

1. Make sure the cursor is at the *Step Number* field, and follow the next steps.

## 7-36 Learning More Features

<b>Action</b>	<b>Explanation</b>
Enter 8	The cursor moves to the action field.
Enter D	This indicates that you wish to delete the current step.
Press <b>Return</b>	Step 8 is deleted from the logic block, and the steps below it are moved up.

A message is then displayed at the bottom of the screen, prompting you to press the **Commit Data** key to make the deletion permanent. The D action blanks out the step from the screen. However, the step is not actually deleted from the logic block until you press **Commit Data**.

1. Press the **Commit Data** function key now.

### **Modifying a Step**

You can now change an existing step. When you entered step 7, you entered the ENTER command as it should finally appear, but this did not point to the correct step at the time. The adjustments HP ALLBASE/4GL made to this step number when you deleted step 8 mean that the ENTER command now refers to the wrong step.

You now need to change this command so that, once again, it refers to the correct step.

1. Position the cursor at the *Step Number* field and follow these steps.

Action	Explanation
Enter 7.	The cursor moves to the action field.
Press <b>Return</b>	Since the default action for an existing step is <i>Change</i> , the current contents of the step are displayed in an IF window and the cursor is positioned in the first field of the window.
Press <b>Tab</b> seven times	This moves the cursor through the fields to the THEN field on the screen.
Type 10 over the top of the number 9	This alters the step so that if both test conditions in the IF command are true, processing control skips to step 10.
Press <b>Commit Data</b>	Commit the step and HP ALLBASE/4GL will display the modified command.

- Position the cursor on the *Action* field, regardless of step number, and display the whole process on the screen.

Action	Notes
Enter L	The <i>L</i> action <i>lists</i> the current logic block to the screen. When you press <b>Return</b> , HP ALLBASE/4GL clears the screen and lists the entire process. Any lines that are too long to fit on the screen will be wrapped around to the next line.
Press <b>Return</b>	This returns you to the process details screen.

- Press the **Generate Process** function key to generate the modified *product\_proc* process.

If you have entered the process exactly as it's listed at the beginning of this section, you won't receive any generate error messages. If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error.

This concludes this lesson. If you are not modifying an HP TurboIMAGE/iX application, turn to the Summary section.

## 7-38 Learning More Features



---

## Modifying HP TurboIMAGE/iX Based Applications

The modified *product\_proc* process reads like this:

```
1 DM IMAGE *MODE *MODLOCK :D-traindb
2 MODE *WRITE product
3 MOVE C-no_record V-product_status
4 SCREEN product_scrn
5 IF V-mode = C-add THEN VISIT add_product
6 IF V-mode = C-modify THEN VISIT modify_product
7 IF V-mode = C-delete THEN VISIT delete_product
8 IF *IOSTATUS <> "00000" & *IOSTATUS <> N-record_not_found
   THEN ENTER 11
9 DM IMAGE *UNLOCK :D-traindb
10 ENTER 4
11 MESSAGE file_error
12 SCREEN main
```

The following paragraphs explain what each step in the process does.

- |                  |   |
|------------------|---|
| Line 1           | This step sets the database mode in which the application accesses the <i>traindb</i> database.   |
| Line 2           | This step sets the HP ALLBASE/4GL access mode for the product data set.   |
| Line 3           | This step initializes the value of the variable <i>V-product_status</i> when the screen is first displayed.   |
| Line 4           | This step displays the <i>product_scrn</i> data screen. The same screen is used for each mode. The setting of the <i>mode</i> variable determines how the process updates the files when the user commits the screen. |
| Lines 5, 6 and 7 | The IF commands test the update mode, and then invoke the appropriate file update function using the VISIT command.   |
| Line 8           | Values of 00000 or 19111 in *IOSTATUS indicate that all file transactions occurred without error, or that the record could not be found.  |

If the file manager return status is neither of these values, a file error has occurred during the update functions. In this case, control passes to the error handling logic starting at step 11.

- Line 9 This step releases any locks held by this process on the *traindb* database. Locks are set in the functions called to add, modify, or delete products.
- Line 10 This step returns control to step 4 to display the *product\_scrn* screen again.
- Line 11 Control only passes to this step if an error is detected at step 6. This step displays a message to warn the user that an error has occurred and the current product information may be corrupted.
- Line 12 This step returns the user to the *main* menu. Displaying a menu is one technique you can use to recover from serious, but non-fatal error conditions. When HP ALLBASE/4GL displays a menu, all current activity in the application ceases. This means that the application returns to a known condition. To continue from the *main* menu, the user must select a menu action. Any actions that can update data files initiate a process, so all file record and screen buffers are cleared automatically. You don't need to release the DM IMAGE \*LOCK in this situation, because all HP TurboIMAGE/iX locks are released automatically when a process exits.

## Modifying Logic Blocks

This part of the lesson shows you how to use facilities that allow you to modify logic blocks. You use the same methods to modify processes and functions.

### To insert a step:

1. Call up the process details screen and follow these steps:

## 7-40 Learning More Features

Action	Explanation
Enter <code>product_proc</code>	This displays the <i>product_proc</i> process in its current state and places the cursor in the step number field.
Enter <code>3</code>	The cursor moves to the action field.
Enter <code>I</code>	This indicates that you want to insert a step before step number 3. The current steps move down one line, and step 3 becomes blank. Then the cursor moves to the command field.
Enter <code>MOVE</code>	This displays an open window. The <code>MOVE</code> command is used to copy the contents of a field to another field.
Enter <code>C-no_record</code> <code>V-product_status</code>	

#### To complete the window:

The `MOVE` command is now complete.

1. Press the `Commit Data` function key to commit it.

HP ALLBASE/4GL clears the window and displays the new command at step 3. The cursor returns to the *Step Number* field and the step number is incremented to 4.

#### Step Number References

Read the last line of the process. Before you inserted a new step, this logic step was `ENTER 3`. Now it is `ENTER 4`.

HP ALLBASE/4GL has automatically incremented the step number reference as the `SCREEN` command has moved down from step 3 to step 4. HP ALLBASE/4GL updates step number references when you insert or delete lines from a logic block.

## Using the IF Window

The next command that you need to insert is step 5. This contains an IF command.

### To use the IF window:

1. Insert a new step 5, and enter IF in the *Command* field.

This displays the IF command window. The fields on this window are explained in more detail below.

2. Enter the appropriate information for each field.

Field	Entry	Explanation
Condition 1		The first three fields of the IF window establish the first condition that the IF command tests. In many cases it is the only condition.
Data Name 1	V-mode	The name of the object to be tested. In this case, the command tests the variable <i>mode</i> , which contains one of the four update modes; add, modify, delete or review.
IF Test	=	The test to be performed. In this case the test is whether or not the content of the <i>mode</i> variable equals the value in the <i>Data Name 2</i> field.
Data Name 2	C-add	Where an IF Test compares two operands with a relational operator such as "=", "<", ">", or "<>", you must enter the name of the second operand in this field.  In this case, the <i>mode</i> variable will be set to the contents of the <i>add</i> alphanumeric constant if the user selected <i>Add</i> on the main menu. The IF command tests to see whether this is the current value of V-mode.

## 7-42 Learning More Features

Field	Entry	Explanation
<b>AND/OR</b>	Leave this field blank.	<p>The IF command can test two conditions and combine the results using a logical AND or a logical OR connective. The symbols <b>&amp;</b> and <b> </b> represent the AND and OR connectives respectively.</p> <p>When you leave this field blank, the cursor skips the next three fields.</p>
<b>THEN Statement</b>	VISIT add_product	<p>When you leave the AND/OR field blank and press <b>(Return)</b>, a <i>THEN</i> prompt appears.</p> <p>The cursor is placed in the field to the right of this prompt. This is where you enter the commands to be performed when the test is resolved as true. You can execute any command at this stage except IFLOOP, SELECT, or another IF statement.</p> <p>You can specify multiple commands by separating the commands with a semicolon. For example, you could enter a string of commands in this field like this:</p> <p><b>MESSAGE</b> added_ok ; <b>ENTER</b> 10</p>
<b>ELSE Statement</b>	Leave field blank.	<p>After you enter the THEN statement, the window displays an <i>ELSE</i> prompt and positions the cursor in the field beside the prompt. This is where you enter the commands to be performed if either test is resolved as false. This entry follows the same format as the THEN statement. However, it is not a required field. If you don't enter any ELSE commands, control passes to the next step in the process or function if the test is resolved as false.</p>

## Learning More Features 7-43

### To complete the window and continue:

The IF command is now complete.

1. Press the **Commit Data** function key to commit it.

HP ALLBASE/4GL clears the window and displays the new command at step 5. The cursor returns to the *Step Number* field and the step number is incremented to 6.

2. Now insert steps 6 and 7, which are very similar to the step you just inserted. These two steps also call up functions to update the *product* file.

The steps are shown below.

```
6 IF V-mode = C-modify THEN VISIT modify_product
```

```
7 IF V-mode = C-delete THEN VISIT delete_product
```

At the moment, there are only three update functions available, and the code is short and simple. However, in later lessons, you will introduce more functionality. Instead of adding a lot of logic commands, you will convert these decision logic lines into one logic command that calls a decision table. Decision tables allow you to define a series of actions to be performed as the result of the outcome of a complex series of conditional tests or questions.

Your process should now look like this:

```
1 DM IMAGE MODE *MODLOCK :D-traindb
2 MODE *WRITE
3 MOVE C-no_record V-product_status
4 SCREEN product_scrn
5 IF V-mode = C-add THEN VISIT add_product
6 IF V-mode = C-modify THEN VISIT modify_product
7 IF V-mode = C-delete THEN VISIT delete_product
8 FILE *WRITE product
9 DM IMAGE *UNLOCK :D-traindb
10 ENTER 4
```

## 7-44 Learning More Features

### Continuing the Change

The next task is to insert a new step 8. This step also uses the IF command, but this time, two sets of conditions are tested.

1. Insert step 8 now, using the following details for each field.

Field	Entry	Explanation
<b>Command</b>	IF	
<b>Data Name 1</b>	*IOSTATUS	In this case, the command tests the HP ALLBASE/4GL communication field *IOSTATUS, which contains either zero, or the error message number returned by the HP ALLBASE/4GL data manager after an access to an application data file.
<b>IF Test</b>	<>	
<b>Data Name 2</b>	"00000"	The value "00000" will be the content of *IOSTATUS if no file errors have occurred. The IF command tests to see whether this is the current value of *IOSTATUS.
<b>AND/OR</b>	&	This represents AND.  * Enter the following information in the appropriate fields:
<b>Condition 2</b>		The next three fields establish a second condition that the IF command tests.
<b>Data Name 1</b>	IOSTATUS	
<b>IF Test</b>	<>	
<b>Data Name 2</b>	N- record_not_found	
<b>THEN Statement</b>	ENTER 11	When you complete the condition fields and press <b>(Return)</b> , a <i>THEN</i> prompt appears. Enter the command shown.

### To complete the window:

The IF command is now complete.

1. Press the `Commit Data` function key to commit it.

### Adding Commands

The following commands can be added to the end of the logic block at steps 12 and 13, respectively.

1. Add the commands as shown below.

12 **MESSAGE** file\_error

13 **SCREEN** main

Both of these logic commands use open windows for you to enter the logic command details.

### Deleting a Step

You can now delete an existing step. We wish to delete the FILE step.

1. Make sure the cursor is at the *Step Number* field, and follow these steps.

Action	Explanation
Enter 9	The cursor moves to the action field.
Enter D	This indicates that you wish to delete the current step.
Press <code>(Return)</code>	Step 9 is deleted from the logic block, and the steps below it are moved up.

A message is then displayed at the bottom of the screen, prompting you to press the `Commit Data` key to make the deletion permanent. The D action blanks out the step from the screen. However, the step is not actually deleted from the logic block until you press `Commit Data`.

2. Press the `Commit Data` function key now.

### Modifying a Step

You can now change an existing step. When you entered step 8, you entered the ENTER command as it should finally appear, but this did not point to the

## 7-46 Learning More Features



correct step at the time. The adjustments HP ALLBASE/4GL made to this step number when you deleted step 9 mean that the ENTER command now refers to the wrong step.

You now need to change this command so that, once again, it refers to the correct step.

1. Position the cursor at the *Step Number* field and follow these steps.

Action	Explanation
Enter 8.	The cursor moves to the action field.
Press <b>Return</b>	Since the default action for an existing step is <i>Change</i> , the current contents of the step are displayed in an IF window and the cursor is positioned in the first field of the window.
Press <b>Tab</b> seven times	This moves the cursor through the fields to the THEN field on the screen.
Type 11 over the number 10	This alters the step so that if both test conditions in the IF command are true, processing control skips to step 11.
Press <b>Commit Data</b>	Commit the step and HP ALLBASE/4GL will display the modified command.

2. Position the cursor on the *Action* field, regardless of step number, and display the whole process on the screen.

Action	Notes
Enter L	The L action <i>lists</i> the current logic block to the screen. When you press <b>Return</b> , HP ALLBASE/4GL clears the screen and lists the entire process. Any lines that are too long to fit on the screen will be wrapped around to the next line.
Press <b>Return</b>	This returns you to the process details screen.

3. Press the **Generate Process** function key to generate the modified *product\_proc* process.

If you have entered the process exactly as it's listed at the beginning of this section, you won't receive any generate error messages. If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error.

---

## Summary

In this lesson you used the process details screen to modify a process logic block.

The process details screen allows you to do the following:

- Modify an existing logic step.
- Insert a new logic step.
- Delete a logic step.
- Display the entire logic block.

The process details screen automatically changes references to other steps in commands such as ENTER if you add or delete steps.

The function details screen provides exactly the same facilities for modifying function logic blocks.

## 7-48 Learning More Features

---

## Lesson 15 - Creating New Logic Blocks

The enhanced *training* application requires a number of new logic blocks. First there are four functions called from the menu, which set the appropriate update mode.

These functions all call the same process, which then controls the display of the product screen and updating of the product file. The process calls a number of functions which perform the appropriate update, depending on the update mode.

In addition, the data screen needs a function associated with the product number field on the product screen.

---

### Objectives

When you have completed this lesson, you will have learned how to create these functions and will have learned some new logic commands.

---

### The Mode Functions

Three of the actions called from the main menu are functions. These functions move the appropriate update mode to the *mode* variable and then call the *product\_proc* process.

1. Create and generate the following functions, using the function header screen and the function detail screen:

- a. **Function** - *prod\_add*

```
1 MOVE C-add V-mode  
2 PROCEED product_proc
```

b. **Function** - *prod\_modify*

- 1 **MOVE** C-modify V-mode
- 2 **PROCEED** product\_proc

c. **Function** - *prod\_delete*

- 1 **MOVE** C-delete V-mode
- 2 **PROCEED** product\_proc

The **PROCEED** statements in these functions invoke the *product\_proc* process. Since any current function terminates when a process is started, these functions don't need an **EXIT** command.

---

## Screen Field Logic

You can now modify the *product\_key\_read* function attached to the *Product Number* field on the screen and window. Previously, this function retrieved a record and refreshed the *product\_scrn* screen. Now you will add extra logic to set a variable, and to perform error testing.

**Function** - *product\_key\_read*

This function is called as an after function on the *product\_no* field on the *product\_scrn* screen. It reads the *product* file to retrieve a record matching the product number entered by the user.

If you are developing the KSAM based application read on below to modify the function.

If you are developing the HP ALLBASE/SQL based application, refer to “Modifying HP ALLBASE/SQL Based Functions”.

If you are developing the HP TurboIMAGE/iX based application, refer to “HP TurboIMAGE/iX Based Functions”.

## 7-50 Learning More Features

## Modifying KSAM Based Functions

Using the logic block modification methods you learned when modifying the *product\_proc* process, you can now modify the *product\_key\_read* function.

### To modify a function:

1. Call up the original function with the function details screen
2. Modify the function
3. Generate the function.

If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error. The following lines show what the function will look like after modification. Use the notes that follow to guide you through the modification.

```
1 FILE *READ product *KEY=* ; ENTER 5
2 SHOW *REFRESH
3 MOVE C-record V-product_status
4 EXIT
5 IF *IOSTATUS <> N-record_not_found
   THEN MESSAGE file_error ; SERIES 6 8 ; EXIT
6 FILE *BUFFER product
7 CLEAR *MAP S-description S-lead_time
8 MOVE C-no_record V-product_status
9 IF V-mode <> C-add THEN MESSAGE no_product ELSE
   MESSAGE add_prod; TIE 2
10 EXIT
```

The following notes describe the operation of the function.

- |        |  |
|--------|--|
| Line 1 | This step reads the <i>product</i> file for a record with a key equal to the value in the current screen field. If an error occurs during the file access, control passes to step 5. |
| Line 2 | A record has been found, so the product screen data fields are refreshed with the contents of the product file record.   |

- Line 3           A product record has been found so the *product\_status* variable is set to reflect this. Other parts of the application use this variable to determine if the current product exists.
- Line 4           The function exits.
- Line 5           This test is performed if the file read operation in step 1 fails. If the error is not the standard *record\_not\_found*, then a number of actions occur. First the *file\_error* message is displayed. Then the SERIES command is executed.
- The SERIES command performs steps 6 to 8 and then returns to the next step. This is the EXIT command that follows the SERIES command.
- Line 6           This command clears the product file record buffer.
- Line 7           This command clears the screen fields from the product *Description* field to the product *Lead Time* field.
- Line 8           A product record has not been found so the *product\_status* variable is set to reflect this.
- Line 9           This step displays a message that depends on the current update mode. If the current mode is *add*, this step displays the message *add\_prod*. Then the TIE command specifies that field number 2 is the next field to be processed.
- If the user has not selected *add* mode, and a product record has not been found, this step displays the message *prod\_not\_exist*.
- Line 10          The function exits at this point.

Now you will create the functions that actually manipulate the *product* file data. Turn to File Manipulation Functions.

## 7-52 Learning More Features

## Modifying HP ALLBASE/SQL Based Functions

Using the logic block modification methods you learned when modifying the *product\_proc* process, you can now modify the *product\_key\_read* function.

### To modify a function:

1. Call up the original function with the function details screen.
2. Modify the function.
3. Generate the function.

If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error.

```
1 SQL find_prod
2 FILE *NEXT product ; ENTER 6
3 SHOW *REFRESH S-product_no S-lead_time
4 MOVE C-record V-product_status
5 EXIT
6 IF *IOSTATUS <> N-end_of_file
   THEN MESSAGE file_error ; SERIES 7 9 ; EXIT
7 FILE *BUFFER product
8 CLEAR *MAP S-description S-lead_time
9 MOVE C-no_record V-product_status
10 IF V-mode <> C-add THEN MESSAGE no_product
    ELSE MESSAGE add_prod; TIE 2
11 EXIT
```

The following notes describe the operation of the function.

- |        |  |
|--------|--|
| Line 1 | This step calls the SQL logic block <i>find_prod</i> , which declares and opens a cursor on the <i>product</i> table.                        |
| Line 2 | This step retrieves the first record of the active set for the cursor. If an error occurs during the table access, control passes to step 6. |
| Line 3 | A record has been found, so the <i>product_scrn</i> screen data fields are refreshed with the contents of the product file record.           |

- Line 4            A product record has been found so the *product\_status* variable is set to reflect this. Other parts of the application use this variable to determine if the current product exists.
- Line 5            The function exits.
- Line 6            This test is performed if the FILE \*NEXT operation in step 2 fails. If the error is not the standard *end\_of\_file*, then a number of actions occur. First the *file\_error* message is displayed. Then the SERIES command is executed.
- The SERIES command performs steps 7 to 9 and then returns to the next step. This is the EXIT command that follows the SERIES command.
- Line 7            This command clears the product file record buffer.
- Line 8            This command clears the screen fields from the *Description* field to the *Lead Time* field.
- Line 9            A product record has not been found so the *product\_status* variable is set to reflect this.
- Line 10           This step displays a message that depends on the current update mode. If the current mode is *add*, this step displays the message *add\_prod*. Then the TIE command specifies that field number 2 is the next field to be processed.
- If the user has not selected *add* mode, and a product record has not been found, this step displays the message *no product*.
- Line 11           The function exits at this point.

The *find\_prod* SQL logic should look like this:

```
SELECT :product FROM sqlgrp.product
  WHERE product_no = :S-product_no.product_scrn
  FOR UPDATE OF description, supplier_no, lead_time;
```

The FOR UPDATE OF clause in this SELECT command specifies the columns of the table that are being updated. This is required so that it is possible to update and delete records.

## 7-54 Learning More Features



Now you will create the functions that actually manipulate the *product* file data. Turn to File Manipulation Functions.

## Modifying HP TurboIMAGE/iX Based Functions

Using the logic block modification methods you learned when modifying the *product\_proc* process, you can now modify the *product\_key\_read* function.

### To modify a function:

1. Call up the original function with the function details screen.
2. Modify the function.
3. Generate the function.

If you receive any generate errors, check that the storage items you defined previously are correctly named. This is the most likely source of error.

```
1 DM IMAGE *LOCK :D-traindb :R-product
2 FILE *READ product *KEY=* ; ENTER 6
3 SHOW *REFRESH S-product_no S-lead_time
4 MOVE C-record V-product_status
5 EXIT
6 IF *IOSTATUS <> N-record_not_found
   THEN MESSAGE file_error ; SERIES 7 9 ; EXIT
7 FILE *BUFFER product
8 CLEAR *MAP S-description S-lead_time
9 MOVE C-no_record V-product_status
10 IF V-mode <> C-add THEN MESSAGE no_product
    ELSE MESSAGE add_prod; TIE 2
11 EXIT
```

The following notes describe the operation of the function.

- |        |  |
|--------|--|
| Line 1 | This step locks all records in the <i>product</i> data set.  |
| Line 2 | This step reads the <i>product</i> file for a record with a key equal to the value in the current screen field. If an error occurs during the file access, control passes to step 6. |

- Line 3            A record has been found, so the product screen data fields are refreshed with the contents of the product file record.
- Line 4            A product record has been found so the *product\_status* variable is set to reflect this. Other parts of the application use this variable to determine if the current product exists.
- Line 5            The function exits.
- Line 6            This test is performed if the file read operation in step 2 fails. If the error is not the standard *record\_not\_found*, then a number of actions occur. First the *file\_error* message is displayed. Then the SERIES command is executed.
- The SERIES command performs steps 7 to 9 and then returns to the next step. This is the EXIT command that follows the SERIES command.
- Line 7            This command clears the product file record buffer.
- Line 8            This command clears the screen fields from the product *Description* field to the product *Lead Time* field.
- Line 9            A product record has not been found so the *product\_status* variable is set to reflect this.
- Line 10           This step displays a message that depends on the current update mode. If the current mode is *add*, this step displays the message *add\_prod*. Then the TIE command specifies that field number 2 is the next field to be processed.
- If the user has not selected *add* mode, and a product record has not been found, this step displays the message *prod\_not\_exist*.
- Line 11           The function exits at this point.

Now you will create the functions that actually manipulate the *product* file data.

## 7-56 Learning More Features

---

## File Manipulation Functions

The functions for file manipulation differ according to the type of application you are developing.

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, read on below.

If you are developing the HP ALLBASE/SQL based application, refer to “HP ALLBASE/SQL Based File Manipulation Functions”.

### KSAM Based and HP TurboIMAGE/iX Based File Manipulation Functions

You will create three file manipulation functions to let users add, modify, and delete records in the *product* file.

#### The Add Function

The function that adds new records to the *product* file is called by the *product\_proc* process.

#### To create the add\_product function:

1. Create the *add\_product* function using the lines shown below.
2. Generate the function.

**Function** - *add\_product*

```
1 IF V-product_status = C-record THEN
  MESSAGE prod_add_fail ; EXIT
2 FILE *INSERT product ; ENTER 6
3 MOVE C-record V-product_status
4 MESSAGE prod_add_ok
5 EXIT
6 MESSAGE file_error
7 EXIT
```

**What This Function Does**

- Line 1            This step checks to see if the record to be added already exists. If it does exist, a warning message is displayed and the function exits.
- Line 2            If the record does not exist, control passes to step 2 to add the record to the file. Step 2 performs an INSERT operation to add a new record.
- Lines 3, 4 and 5    Steps 3 to 5 are executed after a successful insert operation. They set the current status value, display a message to confirm the successful addition of the product record, and then exit the function.
- Line 6            If the write operation in step 2 fails, an error message is displayed by step 6 before the function exits.

### **The Modify Function**

The function that modifies the product records is similar to the function that adds records to the files. The function is quite straightforward, so the following paragraphs simply list the function.

1. Create the *modify\_product* function.
2. Generate the function.

**Function** - *modify\_product*

```

1 IF V-product_status = C-no_record THEN
    MESSAGE no_product ; EXIT
2 FILE *WRITE product ; ENTER 5
3 MESSAGE prod_modify_ok
4 EXIT
5 MESSAGE file_error
6 EXIT

```

### **The Delete Function**

1. Create the *delete\_product* function.
2. Generate the function.

## **7-58 Learning More Features**

### Function - *delete\_product*

```
1 IF V-product_status = C-no_record THEN
  MESSAGE no_product ; EXIT
2 FILE *DELETE product ; ENTER 6
3 MESSAGE prod_delete_ok
4 MOVE C-no_record V-product_status
5 EXIT
6 MESSAGE prod_delete_fail
7 EXIT
```

#### What This Function Does

Line 1 Step 1 ensures that a product record is current. If no product record is current, the function exits immediately. In this case, control returns to the *product\_proc* process. You may want to look back at the *product\_proc* process to see what occurs next.

Line 2 Step 2 performs a DELETE operation to delete the product record is deleted. If a file error is detected in this step, control passes to step 6.

Lines 3,4 and 5 Steps 3 to 5 display a confirmation message for the user and set the *product\_status* variable. The function then exits.

Lines 6 and 7 Control passes to steps 6 and 7 if a file error is detected in step 2. These steps display a message, and then exit.

The following pages describe the file manipulation logic blocks required for the HP ALLBASE/SQL based application. If you are not developing an ALLBASE/SQL application, turn to “The Next Step” to continue developing your application.

## HP ALLBASE/SQL Based File Manipulation Functions

### The Add Function

The function that adds new records to the *product* file is called by the *product\_proc* process.

1. Create the *add\_product* function.

**Learning More Features 7-59**

2. Generate the function.

**Function** - *add\_product*

```
1 IF V-product_status = C-record THEN
    MESSAGE prod_add_fail ; EXIT
2 FILE *INSERT product ; ENTER 6
3 MOVE C-record V-product_status
4 MESSAGE prod_add_ok
5 EXIT
6 MESSAGE file_error
7 EXIT
```

#### **What This Function Does**

- Line 1            Step 1 checks to see if the record to be added already exists. If it does exist, a warning message is displayed and the function exits. If the record does not exist, control passes to step 2 to add the record to the file.
- Line 2            Step 2 contains a FILE \*INSERT command to insert a new record into the *product* table. The FILE \*INSERT command inserts the current buffer contents into the table as a new row.
- The FILE \*INSERT command is equivalent to the following SQL command:
- ```
INSERT INTO sqlgrp.product
VALUES (:F-product_no.product,
       :F-description.product,
       :F-supplier_no.product,
       :F-lead_time.product);
```
- Lines 3, 4 and 5    Steps 3 to 5 are executed after a successful insert operation. They set the current status value, display a message to confirm the successful addition of the product record, and then exit the function.
- Line 6            If the insert operation in step 2 fails, an error message is displayed by step 6 before the function exits.

#### **7-60 Learning More Features**

## The Modify Function

The function that modifies the product records is similar to the function that adds records to the files. The function is quite straightforward, so the following paragraphs simply list the function.

1. Create the *modify\_product* function.
2. Generate the function.

**Function** - *modify\_product*

```
1 IF V-product_status = C-no_record THEN  
    MESSAGE no_product; EXIT  
2 SQL modify_product ; ENTER 5  
3 MESSAGE prod_modify_ok  
4 EXIT  
5 MESSAGE file_error  
6 EXIT
```

This function calls the SQL logic block *modify\_product*.

**SQL Logic Block** - *modify\_product*.

This SQL logic block modifies a row in the *product* table.

1. Create the *modify\_product* logic block.
2. Generate the logic block.

```

UPDATE sqlgrp.product SET
    description = :F-description.product,
    supplier_no = :F-supplier_no.product,
    lead-time   = :F-lead_time.product
WHERE CURRENT OF find_prod;

```

This SQL logic block uses a cursor to update the *product* table. The name of the cursor is *find\_prod*. The function *product\_key\_read* declares and opens this cursor by calling the SQL logic block *find\_prod*. Note that the cursor name is the name of the SQL logic block containing the SELECT command that declares and opens the cursor.

### The Delete Function

1. Create the *delete\_product* function.
2. Generate the function.

**Function** - *delete\_product*

```

1 IF V-product_status = C-no_record THEN
    MESSAGE no_product ; EXIT
2 SQL prod_del ; ENTER 6
3 MESSAGE prod_delete_ok
4 MOVE C-no_record V-product_status
5 EXIT
6 MESSAGE prod_delete_fail
7 EXIT

```

### What This Function Does

- |        |                                                                                                                                                                                                                                                                           |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | Step 1 ensures that a product record is current. If no product record is current, the function exits immediately. In this case, control returns to the <i>product_proc</i> process. You may want to look back at the <i>product_proc</i> process to see what occurs next. |
| Line 2 | Step 2 calls the SQL logic block <i>prod_del</i> . This SQL logic block deletes the record from the <i>product</i> table. If a file error is detected in this step, control passes to step 6.                                                                             |

## 7-62 Learning More Features



Lines 3, 4 and 5    Steps 3 to 5 display a confirmation message for the user and set the *product\_status* variable. The function then exits.

Lines 6 and 7    Control passes to steps 6 and 7 if a file error is detected in step 2. These steps display a message, and then exit.

### **SQL Logic Block *prod\_del***

Step 3 of the *delete\_product* function calls the SQL logic block *prod\_del*. This SQL logic block deletes the record from the *product* table.

#### **To delete a record:**

1. Create the following logic block.
2. Generate the logic block.

```
DELETE FROM sqlgrp.product
WHERE product_no = :F-product_no.product;
```

This SQL logic block demonstrates the technique for deleting a record without using a cursor. In this case, the SQL DELETE command uses an explicit search condition to specify the rows to be deleted.

#### **To delete a record using a cursor:**

You can also delete a record using a cursor. To delete an existing record in an HP ALLBASE/SQL table using a cursor, you can use the following procedure:

1. Use a SELECT command in an SQL logic block to declare and open a cursor on the table.
2. Use a FILE \*NEXT command in a function or process to position the cursor on the first row of the active set and retrieve the record into the file buffer.
3. Use a FILE \*DELETE command in a function or process to delete the record.

The SQL logic block containing the SELECT command, and the subsequent FILE commands must be executed in the same process.

---

## The Next Step

The data manipulation components of this application are now complete. With the logic you have created, you should be able to use the application to add, delete, and modify product records according to processing rules for the application.

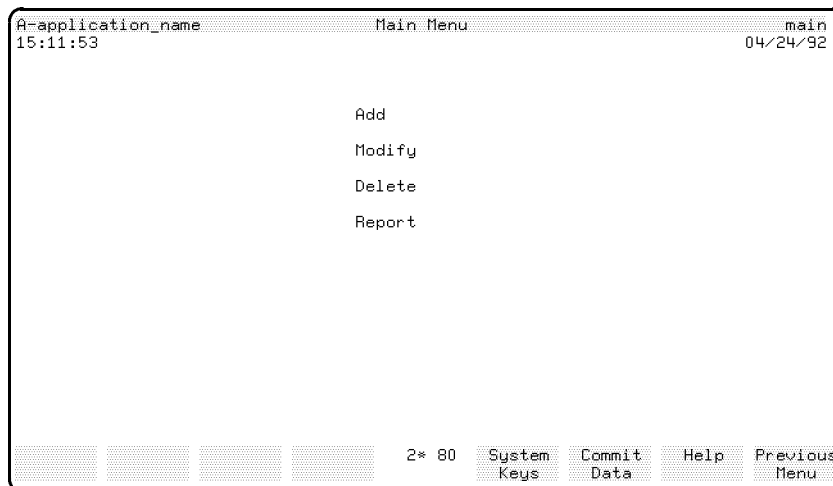
There are a few small things you must do before you can run the application. You must alter the main menu so that you can call up the *product\_scrn* screen in the correct update mode, and you must create the messages that the logic blocks call.

Follow these instructions now to alter the main menu.

### Altering the Main Menu

#### Painting the Screen Again

This is what the main menu will look like when you have repainted it.



**Altered Main Menu for Training Application**

## 7-64 Learning More Features

**To alter the main menu:**

1. Call up the *main* menu in the screen painter.
2. Delete the *Product Details* action item.

Remember that to delete an item, you use the **Delete** function key from the Layout Keys function key set.

3. Change the *Product Report* text to read *Report*.

If this action item doesn't exist on your main menu, refer to chapter 6, lesson 13 and follow the instructions for adding it to the menu.

4. Add the following action items.

Where you place them on the screen is up to you. The screen image above is one suggested format for the menu.

| Type         | Name        | Text   |
|--------------|-------------|--------|
| F (function) | prod_add    | Add    |
| F (function) | prod_modify | Modify |
| F (function) | prod_delete | Delete |

5. When you have defined these items, exit from the painter.

HP ALLBASE/4GL saves the menu as you exit.

---

## Summary

In this lesson you defined a number of functions to complete the procedural logic for the application. You also added action items to the *main* menu to call these functions.

If you are developing the HP ALLBASE/SQL based application, you also defined a number of SQL logic blocks. In combination, commands in SQL logic blocks and commands in logic blocks allow you to retrieve, insert, modify, or delete rows in an HP ALLBASE/SQL table.

At this stage, all of the new logic is in place. However, the messages cannot be displayed when they are required because you haven't created them yet. The next lesson describes the procedure to create the messages used by these functions.

## **7-66 Learning More Features**

---

## Lesson 16 - Creating Messages

---

### Objectives

When you have completed this lesson, you will have learned how to create messages for the enhanced *training* application.

The messages you will create are called by the *product\_proc* process that you have just modified, and the data manipulation functions that you have just created.

---

### Messages

HP ALLBASE/4GL uses five different types of messages. Each has a different effect on the application. The five message types are:

| Type  | Description                                                                              |
|-------|------------------------------------------------------------------------------------------|
| MESS  | Message only.                                                                            |
| QUERY | The user must enter a response.                                                          |
| WARN  | Warning only, message plus terminal bell.                                                |
| ERROR | The user must take action to continue.                                                   |
| ABORT | The application terminates when the user presses <input type="button" value="Return"/> . |

When you select the type of message to use in your application, you must decide what should happen in the application after the message is displayed.

A message telling the user that a record may be modified should be a MESS type message. It is for information purposes only.

A message to confirm that the user wants to delete a record should be a QUERY type message. The user must respond to a query message. You can

test the user's response to a query message to determine the subsequent action taken by the application.

If the user makes an invalid entry, you should use an **ERROR** type message. In this case, the user cannot move off the current field without making a valid entry.

If the field is not crucial, the message could be a **WARN** type message. In this case the message is displayed along with a terminal bell sound. No specific response or action is required from the user.

If the application or the interface to the host operating system detects a fatal error, you may choose to specify an **ABORT** message. In this case, the message text is displayed on the screen, and the application terminates when the user presses **Return**.

This application requires the following 8 messages, referred to in the process and functions you have defined so far.

- This message tells the user that an unexpected file error has occurred. It is an **ERROR** type message:
  - *file\_error*
- This message tells the user that an appropriate action may be taken. It is a **MESS** type message:
  - *add\_prod*
- These messages tell the user that an update action has succeeded. They are **MESS** type messages:
  - *prod\_add\_ok, prod\_modify\_ok, prod\_delete\_ok*
- These messages are used to tell the user that a selected action may not be executed, or has failed. They are **WARN** type messages:
  - *prod\_add\_fail, no\_product, prod\_delete\_fail*

## **7-68 Learning More Features**

---

## Creating a Message

The first message you will create is the *add\_prod* message.

### Menu Path

#### To access the Messages screen:

1. From the main menu, select the *Dictionary* option.
2. Choose *Messages*.

### Description

A message can be constructed from a number of items. Most messages have a literal component. You can also include communication area fields, variables, file record fields, screen fields, and other items. The *HP ALLBASE/4GL Developer Reference Manual* contains a list of all of the items you can use in messages.

```
Developer          Messages          messages
Name              add_prod
Type              MESS (MESS/QUERY/WARN/ERROR/ABORT)
Response Item
Help Name
Message Construction
"New product may be added."
Field Specs  Ranges  Tables  Help Screens  22* 4  System Keys  Commit Data  Help  Previous Menu
```

**Messages Screen**

**To enter the field values:**

| <b>Field</b>                | <b>Entry</b>                | <b>Explanation</b>                                                                                                                                                                                                                                                                 |
|-----------------------------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>                 | add_prod                    | The name of the message within the application.                                                                                                                                                                                                                                    |
| <b>Type</b>                 | m                           | The type code for this message. Note that HP ALLBASE/4GL converts your entry to uppercase, and expands it to the full message type code. The different available codes are listed to the right of this field. You only need to enter the first character of the message type code. |
| <b>Response Item</b>        | Entry not possible.         | This field only applies to QUERY type messages. HP ALLBASE/4GL moves the user's response to the query into the field that you specify here.                                                                                                                                        |
| <b>Help Name</b>            | Entry not possible.         | The name of the help screen that is displayed if the user presses the <b>Help</b> function key while the message is displayed. This option is only valid for QUERY, ERROR, and ABORT messages.                                                                                     |
| <b>Message Construction</b> | "New product may be added." | The content of the message. Note that you must start and finish literal text in messages with double quotes ("). A message can be up to 160 characters long. If it exceeds this length, it is truncated to fit in the message area at the base of the screen.                      |

**To complete the message:**

1. Press the **Commit Data** function key to create this message.

Messages are automatically generated when committed. Once generated, a message is ready to be used in an application.

**Message** - *file\_error*

**7-70 Learning More Features**



**To create another message:**

1. To create the *file\_error* message, follow these steps:

| <b>Action</b>                          | <b>Explanation</b>                                                                                                                                            |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enter <code>file_error</code>          | The name of the message                                                                                                                                       |
| Enter <code>ERROR</code>               | This is an error type message. It requires the user to reenter the data in the current field.                                                                 |
| Press <code>(Return)</code>            | This time you have access to the <i>Help</i> field. Leave it blank for the time being. You can define a help screen for this message at a later time.         |
| Enter <code>"File error, code:"</code> | This is the first part of the message.                                                                                                                        |
| Enter <code>*IOSTATUS</code>           | The current value of the file return code, which is stored in the <code>*IOSTATUS</code> communication area, is included in the message when it is displayed. |
| Enter <code>" has occurred."</code>    | This is the final part of the message.                                                                                                                        |
| Press <code>Commit Data</code>         | The message is generated automatically.                                                                                                                       |

When HP ALLBASE/4GL displays this message, it joins the three components together. You must explicitly enter any spaces that you want to appear between each component. You can define the individual parts of the message on separate lines. If you include more than one item on a line, separate the items with spaces.

**To create the remaining message:**

1. Now create the remainder of the messages.

Only the essential information for each message is listed here.

- a. **Message** - *prod\_add\_ok*

|          |                                            |
|----------|--------------------------------------------|
| Name     | <code>prod_add_ok</code>                   |
| Type     | <code>MESS</code>                          |
| Contents | <code>"Product added successfully."</code> |

b. **Message** - *prod\_modify\_ok*

|          |                                  |
|----------|----------------------------------|
| Name     | prod_modify_ok                   |
| Type     | MESS                             |
| Contents | "Product modified successfully." |

c. **Message** - *prod\_delete\_ok*

|          |                                 |
|----------|---------------------------------|
| Name     | prod_delete_ok                  |
| Type     | MESS                            |
| Contents | "Product deleted successfully." |

d. **Message** - *prod\_add\_fail*

|          |                                                                      |
|----------|----------------------------------------------------------------------|
| Name     | prod_add_fail                                                        |
| Type     | WARN                                                                 |
| Contents | "This product already exists. "<br>"It cannot be added to the file." |

e. **Message** - *no\_product*

|          |                                 |
|----------|---------------------------------|
| Name     | no_product                      |
| Type     | WARN                            |
| Contents | "This product does not exist. " |

f. **Message** - *prod\_delete\_fail*

|          |                                          |
|----------|------------------------------------------|
| Name     | prod_delete_fail                         |
| Type     | WARN                                     |
| Contents | "This product has NOT been<br>deleted. " |

## 7-72 Learning More Features

All the messages necessary at this stage of application development are now complete.

The final step to the expansion of this application is to define a validation range to automatically validate product numbers as they are entered by the user. You will do this in the next lesson.

---

## Summary

There are five types of messages. Each has a different effect on the application. The five message types are:

| Type  | Description                                                                              |
|-------|------------------------------------------------------------------------------------------|
| MESS  | Message only.                                                                            |
| QUERY | The user must enter a response.                                                          |
| WARN  | Warning only, message plus terminal bell.                                                |
| ERROR | The user must take action to continue.                                                   |
| ABORT | The application terminates when the user presses <input type="button" value="Return"/> . |

A message can be constructed from a number of items. Most messages will have a literal component. You can also include communication area fields, variables, file fields, screen fields, and other items.

---

## Lesson 17 - Validation Ranges

---

### Objectives

When you have completed this lesson you will have learned how to define a validation range using the validation ranges screen.

A range defines lower and upper limits for the acceptable values in a field.

---

### Validation Range Definition

Validation ranges are implemented in much the same manner as validation tables. The major difference is that while a range specifies the lower and upper limits to be applied to the field, a table contains a number of discrete values that can be entered in the field.

When a validation range is assigned to a field specification used on a screen, HP ALLBASE/4GL automatically tests user input data against the limits specified by the range. For any given range, the acceptable values include the upper and lower limits, and all values between the limits.

If the entered value is within the upper and lower limits, HP ALLBASE/4GL displays the message associated with the range. Processing then continues in accordance with the type of the message. If you don't define a message in association with the range, HP ALLBASE/4GL displays a system defined ERROR type message by default.

You can also use validation ranges to select records for reporting.

The validation ranges screen allows you to enter the range name, the range itself, and any message to be displayed if the validation fails.

### Menu Path

#### To access the validation ranges screen:

1. From the main menu select the *Dictionary* option.

### 7-74 Learning More Features

2. Select the *Validation Items* option.
3. Choose *Ranges*.

### Description

The range you define here validates the product number entered by the user to ensure that it is a valid product number. Product numbers must begin with the letters XY, and must have a four digit number following the letters. This number cannot be lower than '1000'.

When you have defined the range, you must redefine the *product\_no* field specification to include the validation range. You will also need to regenerate the *product\_scrn* data screen to update the generated record for the screen.

```

Developer: Validation Ranges ranges
-----
Range Name      product_no      Secured  N (Y/N)
Edit Code       U (X/A/U/K/N/S/Q/D/T)
Value  - From   XY1000
Value  - To     XY9999
Number of Decimal Places  0
Message Name    product_no_error

Field Specs.  Tables  Messages  12* 39  System  Commit  Help  Previous
Keys        Data   Menu
  
```

**Validation Ranges Screen**

**To enter the field values:**

| <b>Field</b>                    | <b>Entry</b>     | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Range Name</b>               | product_no       | The name of the range within the application.                                                                                                                                                                                                                                                                                                                                      |
| <b>Secured</b>                  | N                | Indicates whether this item is secured against modification by an unauthorized developer.                                                                                                                                                                                                                                                                                          |
| <b>Edit Code</b>                | U                | The standard edit code field indicating the type of range checking to be performed.<br><br>This code is usually the same as the field to which the range is assigned. If the edit code is <i>N</i> or <i>S</i> then a numeric check is performed. Otherwise the system collating sequence is used. Refer to the <i>HP ALLBASE/4GL Developer Reference Manual</i> for more details. |
| <b>Value - From</b>             | XY1000           | The lower limit of the range.                                                                                                                                                                                                                                                                                                                                                      |
| <b>Value - To</b>               | XY9999           | The upper limit of the range.                                                                                                                                                                                                                                                                                                                                                      |
| <b>Number of Decimal Places</b> | Leave blank.     | For numeric range checking this entry specifies the precision to which the values should be compared.                                                                                                                                                                                                                                                                              |
| <b>Message Name</b>             | product_no_error | The name of the message that is displayed if the value entered by the user does not fit within the specified limits.                                                                                                                                                                                                                                                               |

**To complete this validation range:**

1. Press the **Commit Data** function key to create this validation range.
2. When you have committed the range, include it in the field specification by performing the following steps.

**7-76 Learning More Features**

| Action                                 | Explanation                                                         |
|----------------------------------------|---------------------------------------------------------------------|
| Press <b>Field Specs.</b>              | This displays the field specification screen.                       |
| Enter <code>product_no</code>          | This is the name of the field specification that has to be altered  |
| <b>(Tab)</b> to the <i>Range</i> field | This is the field where you enter the name of the validation range. |
| Enter <code>product_no</code>          | This is the name of the validation range.                           |
| Press <b>Commit Data</b>               | This commits the changes to the field specification.                |

#### To incorporate the new field specification and table:

1. Go to the screen field details screen and call up the *product\_scrn* screen.
2. Call up field number 1.

You will see that the name of the validation range for the field specification has been included automatically.

3. Press the **Generate Screen** function key to incorporate the new field specification and table in the application.

#### To create the `product_no_error` message:

The last thing you need to do is to create the *product\_no\_error* message.

1. Go to the message definition screen.

If you haven't defined any other messages, tables, or ranges since you defined the validation range, the message name should default to *product\_no\_error*.

2. If the default is not *product\_no\_error*, enter the correct name.
3. Assign the message an ERROR type code and enter a message to inform the user that the product number just entered is invalid.
4. Commit the message to generate it.

#### To check on the created messages:

1. Now test your application again and observe how the new items you have defined work together in your application.

---

## Summary

Validation ranges have the following properties:

- Validation ranges can be used to test whether user input data in a screen field is within a specified range of values. Validation ranges can also be used for selecting records for reporting.
- A validation range specification requires an edit code. These are the same as the edit codes that are applied to dictionary field specifications.
- You must specify the starting and finishing values for a validation range. The acceptable values for a validation range include the starting and finishing values.
- If the edit code for a validation range is N or S, HP ALLBASE/4GL performs the validation check according to numeric values. For other edit code types, HP ALLBASE/4GL performs the validation checking on a character by character basis using the system collating sequence.
- The validation range definition allows you to specify the name of a message to be displayed to the user if input data fails the validation check. If you don't specify a message name, HP ALLBASE/4GL displays a system defined default message on validation failure.

This part of the *training* application is now complete. The following chapters introduce you to the module builder feature and how to extend your application even further.

---

## Self Test Questions

**Question 1.** Variables are initialized at the start of an application session. What are they initialized to?

**Question 2.** Which of the following items are storage items?

- Variables.
- Application titles.
- Messages.

### 7-78 Learning More Features



- Validation tables.

**Question 3.**

**KSAM Developers and HP TurboIMAGE/iX Developers Only:**

What does the file error *19111* indicate?

**HP ALLBASE/SQL Developers Only:**

What does the file error *60110* indicate?

**Question 4.** Match the five message types with the five descriptions below.

- A. MESS 1. The application terminates when the user presses **Return**.
- B. QUERY 2. The message is displayed without a terminal bell sound, and no help screen is available.
- C. WARN 3. The user cannot move from the current field when this message is displayed until specific action is taken.
- D. ERROR 4. A user response is required.
- E. ABORT 5. The terminal bell sounds when the message is displayed.

**Question 5.** What logic command can you use to add a record?

**Question 6.** What logic command can you use to modify a record?

**Question 7.** What logic command can you use to delete a record?

**Question 8.** If field validation against a validation range is required, where will the range name be associated with this field?

---

## Answers

**Answer 1.** Alphanumeric variables are initialized to spaces. Numeric variables are initialized to zero.

**Answer 2.** Variables and application titles.

**Answer 3.**

**KSAM Developers and HP TurboIMAGE/iX Developers Only:** 19111 is the file error returned when a record cannot be found.

**HP ALLBASE/SQL Developers Only:** 60110 is the file error returned when the beginning or the end of a table is reached.

**Answer 4.** A2, B4, C5, D3, E1.

**Answer 5.** For KSAM and HP TurboIMAGE/iX files, use FILE \*WRITE filename or FILE \*INSERT filename. For HP ALLBASE/SQL tables, use FILE \*INSERT tablename.

**Answer 6.** For KSAM and HP TurboIMAGE/iX files, use FILE \*WRITE filename. For HP ALLBASE/SQL tables, use an SQL logic block with the following command:

```
UPDATE tablename SET
    field_spec      = :F-field_spec.tablename,
    ....., =
    ....., =
    ....., =
WHERE CURRENT OF sql_block_name;
```

**Answer 7.** For KSAM and HP TurboIMAGE/iX files, use FILE\*DELETE filename.

For HP ALLBASE/SQL tables, use an SQL logic block with the following command:

```
DELETE FROM tablename
WHERE key_field = :F-field_spec.tablename;
```

## 7-80 Learning More Features

You can also delete an HP ALLBASE/SQL record by using a SELECT command, a FILE \*NEXT command, and a FILE \*DELETE command to declare and open a cursor, position the cursor, and delete the record.

**Answer 8.** If the screen field is a dictionary field, then the validation range name is specified in the dictionary during field specification definition.

If the screen field is not a dictionary field (that is, it has not been defined as a field on the field specification screen), the table name can be specified on the screen field details screen.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Learning to Use the Module Builder

---

The HP ALLBASE/4GL module builder allows you to create a working application module very quickly. The module consists of all the application items required to review or update a file. Application items are individual components of the application or module, such as screens, logic blocks, field specifications, and messages.

All the module's application items revolve around a main file and a single screen. Once you have specified the main file, the associated record layout, and the module type (inquiry or maintenance), the module builder automatically creates the resultant screen and logic required to perform maintenance or inquiry tasks on the main file.

When using the module builder, you do not need to create the items individually. That is, you do not need to paint the data screen and complete the field details screen, nor do you need to create any logic blocks, messages or any other application items. All this is performed by the module builder.

The module builder also allows you to link data entered into a field in the resultant screen with existing data in a secondary file. You can use the link feature to validate the entered data with data in a secondary file or to display data from a single record in a secondary file on the resultant screen.

Once the module is built, you can modify the messages, screens and other items built by HP ALLBASE/4GL just as you modify the parts of the application that you create yourself.

---

## Lesson 18 - The Module Builder

In this lesson, you are introduced to the features of the module builder. While working through this lesson you will create the *order* module for the *training* application. The *order* module contains the data for a purchase order, including: the purchase order number, the name of the purchasing officer making the order, the product ordered, the quantity ordered, the cost of the order, the date of the order, and the date of delivery. For simplicity, it is assumed that you can only order one product per order.

To ensure that orders are only placed for existing products, you will link the file to the *product* file for validation. You will also display the product *description* field on the update screen.

This lesson shows you how to use the HP ALLBASE/4GL module builder, and how to use the modules that are created.

---

## Objectives

Upon completing this lesson, you will have learned how to build a simple module. You will have learned how to:

- Create the *order* field specifications, record layout, and file.
- Define the module using the module builder screen.
- Refine the module and link it to the *product* file, using the module details screen.
- Generate the complete module.
- Attach an action item to the *main* menu to run the module.

## 8-2 Learning to Use the Module Builder

---

## Necessary Preparations

Before you can use the module builder you must first define and create the field specifications and the *order* record layout. Then you need to create the *order* file.

The *order* file contains the following fields:

- `order_no`
- `officer_name`
- `product_no`
- `quantity`
- `unit_measure`
- `cost`
- `order_date`
- `delivery_date`

### Defining the Field Specifications

The *product\_no* field has already been created. However, you will now need to create the other dictionary field specifications.

#### To create the field specifications:

1. Create the field specifications for each field listed below.

Only the field entries that differ from the defaults are listed.

- a. **Order Number.** A field for the purchasing order number.

|                  |                 |
|------------------|-----------------|
| Field Spec. Name | <b>order_no</b> |
| Field Length     | <b>5</b>        |
| Edit Code        | <b>N</b>        |
| Storage Type     | <b>I</b>        |
| Blank When Zero  | <b>Y</b>        |

- b. **Officer Name.** A field for the name of the purchasing officer who completed the order.

|                  |                     |
|------------------|---------------------|
| Field Spec. Name | <b>officer_name</b> |
| Field Length     | <b>30</b>           |
| Edit Code        | <b>X</b>            |

- c. **Quantity.** A six digit number used to specify a quantity of a product.

|                  |                 |
|------------------|-----------------|
| Field Spec. Name | <b>quantity</b> |
| Field Length     | <b>6</b>        |
| Edit Code        | <b>N</b>        |
| Storage Type     | <b>I</b>        |
| Blank When Zero  | <b>Y</b>        |

- d. **Unit of Measure.** A four character field indicating the unit of measure that the quantity is expressed in.

|                  |                     |
|------------------|---------------------|
| Field Spec. Name | <b>unit_measure</b> |
| Field Length     | <b>4</b>            |
| Edit Code        | <b>X</b>            |

- e. **Cost.** The cost will be limited to under one billion (\$1000 million) dollars, so the field length will be set to *12*. This allows nine figures, a decimal point, and two decimal places. You specify the two decimal places in the *Decimal Places* field.

#### 8-4 Learning to Use the Module Builder



|                  |             |
|------------------|-------------|
| Field Spec. Name | <b>cost</b> |
| Field Length     | <b>12</b>   |
| Edit Code        | <b>N</b>    |
| Storage Type     | <b>P</b>    |
| Decimal Places   | <b>2</b>    |
| Blank When Zero  | <b>Y</b>    |

- f. **Order Date.** The date that the order was placed.

To define a date field specification, you must choose the *D* or *T* edit code. Both are date fields, but the *T* edit code defaults to the current date. The order date is usually going to be the date that the details are entered by the end user, so, to make the end user's task a little quicker, the order date will use the *T* edit code.

Both date fields allow users to enter the date in either MM/DD/YY format or DD/MM/YY format, depending on the system-wide date format set by the HP ALLBASE/4GL administrator.

|                  |                   |
|------------------|-------------------|
| Field Spec. Name | <b>order_date</b> |
| Field Length     | <b>8</b>          |
| Edit Code        | <b>T</b>          |

- g. **Delivery Date.** The date that the order was delivered.

This time, you will use the *D* edit code for the date, because this field can be left blank until the delivery is made.

|                  |                      |
|------------------|----------------------|
| Field Spec. Name | <b>delivery_date</b> |
| Field Length     | <b>8</b>             |
| Edit Code        | <b>D</b>             |

This completes the definition of the field specifications that are used in the *order* file.

### Creating the Record Layout

One record layout is required, named *order*.

#### To create the record layout:

1. Create the record header, with a suitable description.
2. Create the record layout as specified below.

**Record Layout Name.** *order*

| Field | Field Spec Name | Key # | Duplicates |
|-------|-----------------|-------|------------|
| 1     | order_no        | 1     | N          |
| 2     | officer_name    | 2     | Y          |
| 3     | product_no      | 3     | Y          |
| 4     | quantity        |       |            |
| 5     | unit_measure    |       |            |
| 6     | cost            |       |            |
| 7     | order_date      | 4     | Y          |
| 8     | delivery_date   | 5     | Y          |

3. Generate the record layout once you have defined it.

### Creating the File Specification

Now that you have generated the *order* record layout, you are ready to create the *order* file, table, or data set. The details for each data manager are provided below.

1. Complete the file definition for the data manager you are using.

#### KSAM Based Applications.

- Define and create the *order* file, which contains all the order records.

## 8-6 Learning to Use the Module Builder

|                |              |
|----------------|--------------|
| File Name      | <b>order</b> |
| File Type      | <b>I</b>     |
| External Name  | <b>ORDER</b> |
| Default Record | <b>1</b>     |
| Record Layout  | <b>order</b> |

### **HP ALLBASE/SQL Based Applications.**

- Define and create the *order* table, which contains all the order records.

|                  |               |
|------------------|---------------|
| File Name        | <b>order</b>  |
| File Type        | <b>S</b>      |
| External Name    | <b>ORDER</b>  |
| SQL DBfilesset   | <b>ORDRFS</b> |
| SQL Access Class | <b>1</b>      |
| Record Layout    | <b>order</b>  |

### **HP TurboIMAGE/iX Based Applications.**

- Define the *order* data set, which contains all the order records.

|                |                |
|----------------|----------------|
| File Name      | <b>order</b>   |
| File Type      | <b>D</b>       |
| Database Name  | <b>traindb</b> |
| External Name  | <b>ORDER</b>   |
| Default Record | <b>1</b>       |
| Record Layout  | <b>order</b>   |

This data set is a detail data set. It contains the *product\_no* field, which is an index to the *product* manual master data set. This relationship cannot be specified within HP ALLBASE/4GL; it must be specified in the schema file used to physically create the database.

In addition, each key defined in the *order* record layout should be recorded in an automatic master data set. These may be, but do not need to be, defined in HP ALLBASE/4GL. The *traindb* database created by your administrator already contains the automatic masters required and defines the link between the *order* detail data set and the *product* manual master data set.

---

## Using the Module Builder Screen

You are now ready to use the module builder to build the resultant screen and associated logic needed to allow users to access the *order* file.

To access the module builder screen, choose *Module Builder* from the main menu. HP ALLBASE/4GL displays the module builder screen shown below.

|                    |                                           |                |                 |                |                  |
|--------------------|-------------------------------------------|----------------|-----------------|----------------|------------------|
| Developer          |                                           | Module Builder |                 | module_builder |                  |
| Module Name        | order                                     | Type           | M (I/M)         |                |                  |
| Main Access: File  | order                                     | Record         | order           | Index          | order_no         |
| Include all Fields | <input checked="" type="checkbox"/> (Y/N) |                |                 |                |                  |
| Dict.<br>Menu      | Screens<br>Menu                           | Logic<br>Menu  | Reports<br>Menu | 7* 21          | System<br>Keys   |
|                    |                                           |                |                 |                | Commit<br>Data   |
|                    |                                           |                |                 |                | Help             |
|                    |                                           |                |                 |                | Previous<br>Menu |

**Module Builder Screen**

### Description

You use the module builder screen to specify the essential information required by the module builder to create the *order* module. This screen requires you to specify the main file name, along with a record layout name and an index field

### 8-8 Learning to Use the Module Builder

name. The module uses the specified record layout and index field to access the main file.

**To enter the field values:**

| <b>Field</b>             | <b>Entry</b>                     | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------------------------|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Module Name</b>       | <b>order</b>                     | This name is used by the module builder in the names it gives most application items it creates for the module. The naming conventions used by the module builder are described in the <i>HP ALLBASE/4GL Developer Reference Manual</i> .                                                                                                                                                                                                             |
| <b>Type</b>              | Accept the default of <b>M</b> . | There are two possible module types: maintenance and inquiry. A maintenance type module allows the end user to view and modify the information in the file. An inquiry type module only allows the end user to view the file.                                                                                                                                                                                                                         |
| <b>Main Access: File</b> | <b>order</b>                     | This field specifies the file you defined earlier as the main file to be accessed by the module for viewing or maintenance.                                                                                                                                                                                                                                                                                                                           |
| <b>Record</b>            | Accept the default.              | This is the record layout used to access the file. HP ALLBASE/4GL automatically enters the default record layout.<br><br>This field is bypassed for HP ALLBASE/SQL tables, because each table can only contain one record layout.<br><br>KSAM files and HP TurboIMAGE/iX data sets may contain more than one record layout. If you are developing your application based on either of these data managers, accept the default value of <b>order</b> . |

| Field                     | Entry                                         | Explanation                                                                                                                                                                                                                                                                                                                               |
|---------------------------|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Index</b>              | Accept the default of <code>order_no</code> . | This is the name of the field used to locate an individual record within a file. This value defaults to the primary index for the file.                                                                                                                                                                                                   |
| <b>Include All Fields</b> | Accept the default of <code>Y</code> .        | This tells the module builder to include all fields listed in the record layout in the resultant screen. If you enter <code>N</code> , you must use the module details screen to define which fields from the record layout you want in the resultant screen. The module builder always includes the index field in the resultant screen. |

**To complete and commit the screen:**

1. Press the `Commit Data` function key when you have completed the module builder screen.

HP ALLBASE/4GL displays a message asking you either to specify details on the module details screen or to generate the module.

**Note**



At this point you could generate the module, provided you specified `Y` at the *Include All Fields* prompt. Module builder would attempt to place all the fields in the record layout on the resultant screen, and there would be no links to secondary files. However, in this lesson you will access the module details screen to add the links to the secondary file.

2. Press the `Specify Details` function key to display the module details screen.

**8-10 Learning to Use the Module Builder**

---

## Using the Module Details Screen

You use the module details screen to specify which fields from the record layout are included in the resultant screen. You can also specify which fields in the resultant screen are linked with secondary files. Access to the module details screen, which is shown below, is only possible after you have committed the module builder screen.

| Developer                        |            | Module Builder  |             | module_details |                                |
|----------------------------------|------------|-----------------|-------------|----------------|--------------------------------|
| Module Name                      | order      | Type            | M (I/M)     |                |                                |
| Main Access:                     | File order | Record          | order       | Index          | order_no                       |
| Include all Fields               | Y (Y/N)    |                 |             |                |                                |
| ----- Screen Field Details ----- |            |                 |             |                |                                |
| Sequence Number                  | 1          | Action          | (A/C/D/I/L) |                |                                |
| File                             | order      |                 |             |                |                                |
| Field Spec. Name                 | order_no   | ( )             |             |                |                                |
| On Screen Label                  | order_no   |                 |             |                |                                |
| Type                             | I (I/D)    |                 |             |                |                                |
| Required                         | Y (Y/N)    |                 |             |                |                                |
| Link                             | N (Y/N)    |                 |             |                |                                |
| Validate                         | (Y/N)      |                 |             |                |                                |
| Link to:                         | File       | Record          |             | Index          |                                |
| List Record                      |            | Generate Module | 11* 39      | System Keys    | Commit Data Help Previous Menu |

**Module Details Screen**

### Screen Description

To create a screen, the module builder takes the fields from the record layout and creates corresponding screen fields. With the module details screen, you can change, delete, insert, and add fields to the resultant screen. You can define whether a field is for display or input, and whether it is a field that always requires an entry. You can also alter the field's on screen label, which defaults to the text in the short description field of the field specification.

You also use the module details screen to link a field in the resultant screen with a secondary file field. To build this feature into your application you first specify to which file the field is linked. Then if necessary, you specify the record layout and index field for this file. In the resultant module, there is an after function associated with the link field. This after function looks through

the index field of the secondary file for a record with data matching that in the link field. This is the matching record.

You can then use this match to display data from other fields of the matching record on the resultant screen, or validate the entered data against data in the index field of the secondary file. The secondary file records act the same way as a validation table.

You can only create one-to-one links between a screen field and a secondary file. This means you can link each screen field to only one secondary file, and each secondary file can only be linked to one screen field. You can only access one record at a time in the secondary file. However you can link up to twenty fields with different secondary files.

**To enter the field values for *order\_no*:**

| Field           | Entry               | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sequence Number | Accept the default. | When you commit the <i>Sequence Number</i> field, HP ALLBASE/4GL displays the details for the first screen field, <i>order_no</i> .<br><br>The module builder always includes the index field you specified as the first field in the resultant screen. As this field is used for record retrieval, its function within the resultant screen cannot be modified. Consequently you can only edit the <i>On Screen Label</i> field. |
| Action          | Accept the default. | This action code works in the same manner you encountered when creating logic blocks.                                                                                                                                                                                                                                                                                                                                             |
| On Screen Label | Order Number        | This field defaults to the short description field on the field specification screen. This is the label text displayed next to the <i>order_no</i> input field on the resultant screen.                                                                                                                                                                                                                                           |

**To complete and commit the screen:**

1. Press the **Commit Data** function key.

**8-12 Learning to Use the Module Builder**



**To enter the field values for Officer Name:**

| <b>Field</b>            | <b>Entry</b>        | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Sequence Number</b>  | Accept the default. | Notice that after you commit the <i>Sequence Number</i> field, HP ALLBASE/4GL fills the <i>File</i> , <i>Field Spec. Name</i> , and <i>On Screen Label</i> fields with default values. This screen sequence is defined by the record layout <i>order</i> . HP ALLBASE/4GL uses the record layout for the screen field sequence if you specify Y at the <i>Include All Fields</i> prompt on the module builder screen. |
| <b>Action</b>           | Accept the default. |                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>File</b>             | Accept the default. | The screen field's primary movement field is in the file buffer specified in this field.                                                                                                                                                                                                                                                                                                                              |
| <b>Field Spec. Name</b> | Accept the default. | The data in this field completes specification of the primary movement field for the screen field.                                                                                                                                                                                                                                                                                                                    |
| <b>On Screen Label</b>  | <b>Officer Name</b> |                                                                                                                                                                                                                                                                                                                                                                                                                       |

**To complete and commit the screen:**

1. Press the **Commit Data** function key to complete the specification of details for the *officer\_name* screen field.

**Linking a Field to a Secondary File**

Now link the *product\_no* field to a secondary file.

**To link a field to a secondary file:**

| <b>Field</b>           | <b>Entry</b>                                      | <b>Explanation</b>                                                                                                                                                                                                                                                                                                    |
|------------------------|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Sequence Number</b> | Accept the default.                               | Accept the default values for the <i>Action</i> , <i>File</i> and <i>Field Spec. Name</i> fields.                                                                                                                                                                                                                     |
| <b>On Screen Label</b> | <b>Product Number</b>                             |                                                                                                                                                                                                                                                                                                                       |
| <b>Type</b>            | I                                                 | This indicates that <i>product_no</i> is an input field, not a display field.                                                                                                                                                                                                                                         |
| <b>Required</b>        | Y                                                 | This indicates that <i>product_no</i> is a required field. If the user leaves the field blank, an error message will be displayed.                                                                                                                                                                                    |
| <b>Link</b>            | Y                                                 | This tells the module builder to create an after function to link the data entered by the user into the <i>Product Number</i> field with data in the secondary file named below.                                                                                                                                      |
| <b>Validate</b>        | Y                                                 | This tells the module builder to include, in the after function, logic to validate the data entered into the screen field with the index field of the secondary file. It means that the user cannot enter a product number into the <i>order</i> file if the product number doesn't exist in the <i>product</i> file. |
| <b>Link To: File</b>   | <b>product</b>                                    | This is the name of the secondary file linked to the <i>product_no</i> field.                                                                                                                                                                                                                                         |
| <b>Link To: Record</b> | This field is bypassed for HP ALLBASE/SQL tables. | For KSAM based applications and HP TurboIMAGE/iX based applications, accept the default value <b>product_rcrd</b>                                                                                                                                                                                                     |
| <b>Link To: Index</b>  | Accept the default.                               | The link after function searches through the index field of the secondary file. It retrieves the record where data in the index field matches that entered in the link field.                                                                                                                                         |

**8-14 Learning to Use the Module Builder**

**To complete and commit the screen:**

1. Press the `Commit Data` function key.

**Inserting a Display Field**

This field shows the description of the product. It displays the *description* field of the product record matching the data entered in the link field, *product\_no*.

Any fields displaying data from the secondary file must appear on the screen after the link field, that is, the display fields must have a higher sequence number. This is because the appropriate record must be retrieved before other fields can be displayed. In this application, the product number must be entered before the description of the correct product can be displayed.

**To insert a display field:**

| Field            | Entry                    | Explanation                                                                                                                                                                                                                                   |
|------------------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sequence Number  | 4                        | This is the number of the next field in the resultant screen field sequence. This displays the next field from the <i>order</i> file, which is <i>quantity</i> .                                                                              |
| Action           | I                        | This means that you wish to insert a field before the <i>quantity</i> field, which is currently field number 4. When you press <code>Return</code> , module builder alters the defaults to the <i>product</i> file and default record layout. |
| File             | Accept the default.      |                                                                                                                                                                                                                                               |
| Field Spec. Name | <code>description</code> |                                                                                                                                                                                                                                               |
| On Screen Label  | <code>Description</code> | Notice that because this is a field from a secondary file, it automatically defaults to a display field. You cannot specify any links to this field.                                                                                          |

**To complete the field:**

1. Press the `Commit Data` function key.

**To complete other field labels:**

1. Use the same procedure that you used to enter the details of screen fields 1 and 2 to enter the details of the screen fields listed below.

| Sequence No. | Field Spec. Name | On Screen Label |
|--------------|------------------|-----------------|
| 5            | quantity         | Order Quantity  |
| 6            | unit_measure     | Unit            |
| 7            | cost             | \$              |
| 8            | order_date       | Ordered         |
| 9            | delivery_date    | Delivered       |

---

## Final Steps

You are now ready to perform the final steps required to create the *order* module; generating the module and joining it to the *main* menu. Then you can run the *training* application to investigate the capabilities of the *order* module.

### Module Generation

Press the **Generate Module** function key to create the *order* module.

HP ALLBASE/4GL now works through the process required to build and generate a working module. When complete HP ALLBASE/4GL displays the message *"Build has completed successfully. Please press the Return key to proceed."* When you press **(Return)**, HP ALLBASE/4GL displays the module builder screen.

### The Module Building Process

The process HP ALLBASE/4GL follows to build the *order* module can be summarized into three main steps:

1. Creating the resultant screen and the process that calls it.
2. Creating all other application items.

## 8-16 Learning to Use the Module Builder

3. Generating all the module's items.

For more information, refer to the *HP ALLBASE/4GL Developer Reference Manual*.

## Joining the Module to the Main Menu

Finally you need to add an action item to the *main* menu which accesses the *order* module. Do this now.

1. Use the screen painter to add an action item to the *main* menu which calls the process *mb\_order*.
2. Label the action item *Order Details*.

---

## Further Options

Once the *order* module has been created, you have several further options. You can modify the items created to alter the module's functionality, or you can use items created for the module in other parts of an application.

You treat all the items created by the module builder in the same way as those you create yourself. All module builder items are listed in the catalog display. Modification is quite simple because the templates the module builder uses to create items contain the same HP ALLBASE/4GL features available to you.

Copying items created by module builder to other parts of your application speeds up the creation of an application. An example of where you can use items elsewhere is the after function for the *Product Number* field in the *mb\_order* screen, called *mb\_order\_1\_08*. In the *mb\_order* screen, this after function validates the product number entered against the secondary file *product*. Thus it ensures the user only enters a valid product number into the *mb\_order* screen. You could copy and then modify this function to become an after function to validate the *Product Number* field in the *product\_scrn* screen. In this context, the after function would validate data entered into the *Product Number* field against the file *product*.

---

## Running the Module

You can now test the *order* module by selecting the *Test* option from the developer main menu.

When you select the *Order Details* option from the training application main menu, the following screen is displayed:

```
training                Order Maintenance Screen                mb_order

      Order Number
      Officer Name
      Product Number
      Description
      Order Quantity
      Unit
      $
      Ordered
      Delivered

Mode (Add)  Previous Record  Next Record  More Keys  7  40  System Keys  Commit Data  Help  Previous Menu
```

**Order Maintenance Screen**

## Using the Module

The *order* maintenance module allows you to add new records, and delete or modify existing records in the file. Each action is performed when the module is in the corresponding mode for the action. The current module mode is displayed in the label for the **Mode** function key, located in the bottom left corner of the screen.

The mode indicates the type of change that will be made to the file when you press the **Commit Data** function key.

HP ALLBASE/4GL automatically sets the mode based upon your previous action. If you enter an order number that does not currently exist, the mode will change to *Add*. If the order number does exist, the mode will change to *Modify*.

### 8-18 Learning to Use the Module Builder

You can then change the mode by pressing the **Mode** function key.

This module has three possible modes: add, modify, and delete. (Depending on the type of file, and whether the file records have a unique primary key, some modules may not provide all three modes.) When you first enter the resultant screen, the mode is set to add.

Add some records, then try to modify or delete them. The help screens provided with the module may give you some assistance in using the screen.

Test what happens when you enter an invalid product number. (A valid product number is defined as XY1000 to XY9999.)

### **Retrieving an Existing Record**

File scanning allows you to select the first, next, previous, or last record from the file. All scanning is controlled with four function keys: **Previous Record**, **Next Record** in the main keys set, and **First Record**, and **Last Record** which are displayed if you press the **More Keys** function keys.

Because the *order* record layout allows unique primary key values, you can use keyed access to retrieve a specific record. This is done by entering a value in the key field and pressing **Return**. HP ALLBASE/4GL attempts to locate and display a record with a matching key value. This is a full key read of the main file, and is available whenever the cursor is in the key field.

If a record is located, you can access records with key values adjacent to that entered by pressing the **Previous Record** and **Next Record** function keys. You can then modify the retrieved record, or add a new record if no record is found.

Once you have created a number of records, use the file scanning keys and the keyed access method to display different records.

---

## Summary

In this lesson you created the *order* module using the HP ALLBASE/4GL module builder.

### Necessary Preparations

Before you used the module builder, you had to define and create the field specification names, record layout and the physical file required for the module you created.

### Module Building

To create the module you accessed the module builder screen to define the following:

- The module name, which is used in the naming of application items created by the module builder.
- The module type. You can create maintenance modules, which allow the end user to view, modify, delete, or add information in a file, or inquiry modules, which only allow the user to view information in a file.
- The main file accessed by the module. The screen and associated logic created can only modify one file; the main file. When you specify the main file you must also specify the record layout and index field you are using to access it.
- Whether the module builder automatically places all the fields in the record layout in the resultant screen.

Once you have committed the module builder screen, you can either generate the module or move to the module details screen to refine the module. If you want to generate the module from the module builder screen, you must specify *Y* at the *Include All Fields* prompt.

On the module details screen you performed the following:

- Define which fields from the main file record layout are placed in the resultant screen.

## 8-20 Learning to Use the Module Builder



- Define a link between a field in the resultant screen and a secondary file.
- Use this link to validate the data entered into the link field.
- Use this link to display information from a record in the secondary file on the resultant screen.

When this screen was complete, you generated the module.

### **Joining the Module to the Application**

After the module had been generated you joined it to the rest of your application by adding an action item to the *main* menu. This action item calls the main process generated by the module builder.

### **Running the Module**

When you use application testing to run a module, there are several operating features that you can use:

- A **Mode** function key allows you to change between add, modify, and delete modes.
- The mode changes automatically when you attempt to load a record. If it exists, the module changes to Modify mode. If the record does not exist, the module changes to Add mode.
- Four function keys allow you to move to the first record, last record, previous record, and next record in the file.
- If the file contains a unique primary key, you can use keyed access to find the record matching, or closest to, the primary key value that you enter in the first field on the screen.

### **Conclusion**

This part of the *training* application is now complete.

You may wish to stop reading the guide here, and practise writing your own applications. When you feel that you have understood the material covered so far in this training guide, venture on to the next chapters.

The lessons in the following chapters show you how to develop the items needed to use multiple files in conjunction with an HP ALLBASE/4GL

application. These lessons also introduce some more advanced screen handling techniques.

---

## Self Test Questions

- Question 1.** What items must exist before you can build a module?
- Question 2.** What is the difference between a *T* edit code and a *D* edit code?
- Question 3.** When do you need to complete the module details screen to build a module?
- Question 4.** Can you link a secondary file to more than one screen field?
- Question 5.** What does validation against a secondary file do?
- Question 6.** What is the importance of the name you give the module?
- Question 7.** Can you alter the items that the module builder creates?
- Question 8.** What are the three main steps HP ALLBASE/4GL takes to build a module that you have defined?

---

## Answers

**Answer 1.** A file (with a record layout and field specifications) must exist for you to build a module to maintain or inquire on the file.

**Answer 2.** Both the *T* edit code and the *D* edit codes are date formats. However, fields with the *T* edit code default to the current date. There is no default for the *D* edit code.

**Answer 3.** When you specify **N** in the *Include all Fields* field on the module builder screen, you must complete the module details screen to specify which fields the module should include. You must also use the module details screen if you wish to link fields to a secondary file or if you wish to alter the on screen label of a field.

**Answer 4.** No, each secondary file can only be linked to one screen field.

**Answer 5.** If you validate a field against a secondary file, any value that the user enters in the field must match the field in a record in the secondary file. If no record in the secondary field contains that field value, an error message is displayed. You cannot move off the field until a valid entry is supplied.

**Answer 6.** The name that you give the module is used to name all of the items that the module builder creates. A complete list of the items, and their names, is contained in the *HP ALLBASE/4GL Developer Reference Manual*.

**Answer 7.** Yes, the items that the module builder creates are no different to the items you create.

**Answer 8.** HP ALLBASE/4GL builds the module in three main steps:

1. Create the resultant screen and the process that calls it.
2. Create all the other application items required.
3. Generate all the module application items.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Learning Additional Screen Techniques

---

You have now been introduced to most of the fundamental areas of HP ALLBASE/4GL. This chapter and the following chapters will show you how to extend your application even further. As well as using methods you have already seen, these chapters introduce a number of new techniques and new items.

The style of these lessons differs from the earlier lessons. Each keystroke is no longer fully defined for you. In most cases you're only given a description or diagram of the completed item. It's up to you to use the appropriate facilities to create the item.

This chapter draws your attention to any new techniques that are introduced. In addition, it discusses the purpose of the items you create. We recommend that you take your time going through these lessons. Look back over what you have already done, use the developer menu chart, and if necessary, use the *HP ALLBASE/4GL Developer Reference Manual* for more detailed information about the items being used.

Throughout these lessons, we suggest that you create your own descriptions on screens that have description fields.

---

## About the Enhanced Application

This chapter and the two following chapters describe a substantial extension to the existing *training* application. You will develop the items needed to use multiple files together in an application. This system involves the *product* file and an *option* file.

### Product/Option Application

The application involves a product/option relationship where each product must have at least one option, and can have any number of options related to it.

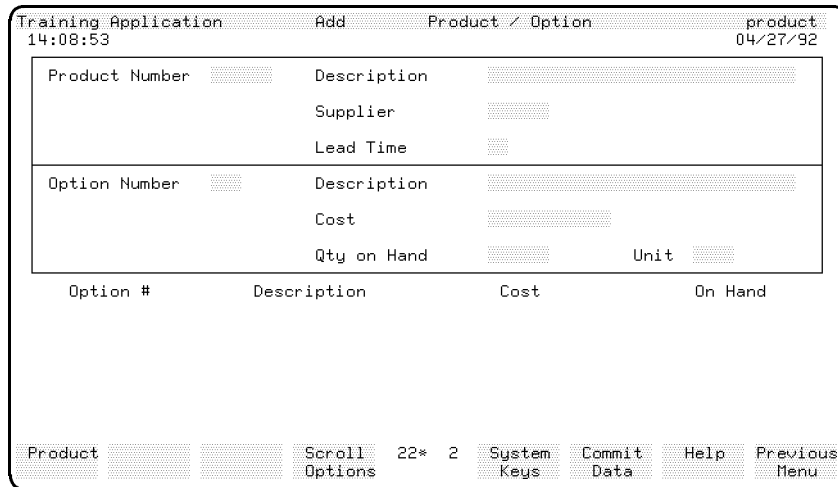
To add the new features to this application, you must create some new screens using windows, scroll areas, and function keys. These additions require a number of new logic blocks as well as a decision table to support them. Finally you will create a new report that uses sort levels and file linkages. You perform these steps in the next three chapters.

As before, the main menu allows the user to select an update mode. The *product\_scrn* data screen is then displayed for each mode.

When necessary, this screen uses a window to allow the user to add, modify, or delete option records.

The screen and window appear as shown here.

## 9-2 Learning Additional Screen Techniques



**product\_scrn Data Screen**

### Function Keys

The data screen and windows used in this part of the application have sets of function keys that allow the user to swap between product maintenance and option maintenance.

### Scrolling Options

At any time when a valid product record has been retrieved, the user can press a **Scroll Options** function key to display the details of the options for the current product. The option details are displayed in a scroll area at the bottom of the screen.

### Reporting

The extensions to the application include a report that lists the details of products and options.

The user can specify a range of products for reporting. For all products in the specified range, the report lists the details of the products and the options. It also calculates the cost of stock for each option, the cost of stock for each product, and the total cost of stock for all products listed.

## **Processing Rules**

The processing rules for the product/option maintenance system are as follows:

- A product may be added or modified any time. When a product is added to the system, a basic option must also be added. This option must have the number “000”.
- A product may only be deleted if there are no options except option number 000 associated with it.
- When a product is deleted, option 000 must also be deleted.
- An option, other than option 000, can be added to an existing product.
- An option may be modified at any time.
- An option, other than option 000, may be deleted at any time.

---

## **Application Structure**

Once again, the application uses the concept of a process calling a screen. Functions associated with fields on the screen perform the necessary file look-up operations. When the user terminates the screen, control returns to the original process which then performs the appropriate file update operations.

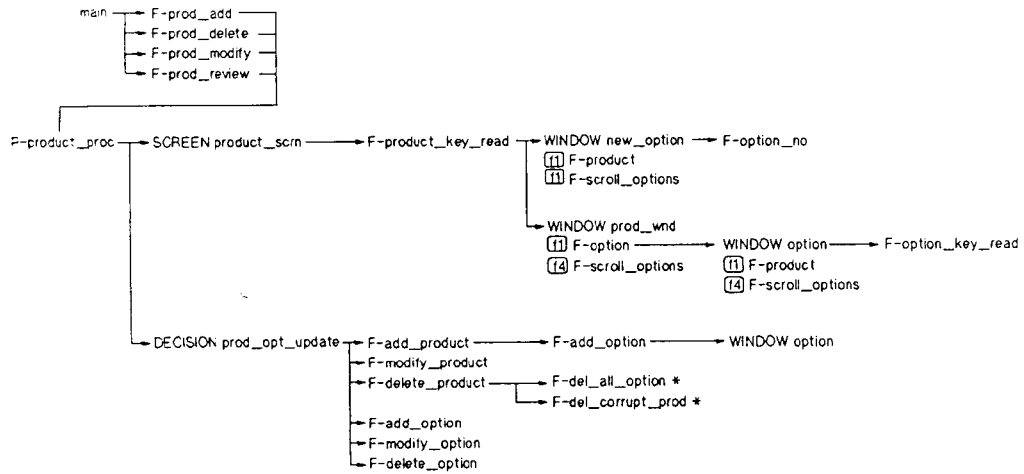
## **Product/Option Application**

In the product/option application, the functions associated with screen fields or screen function keys can also display one of three different windows on the screen. Other functions called by function keys display data in a scroll area on the screen.

The structure of the application can be shown diagrammatically as follows.

## **9-4 Learning Additional Screen Techniques**





\* Indicates that this function is not necessary for HP ALLBASE/SQL based applications

### Diagrammatic Structure of the Enhanced Training Application

On exit from the *product\_scrn* screen, control returns to the *product\_proc* process. The process calls a decision table to determine which function is executed to update the product and option files data.

You will probably want to refer back to this diagram as you define the various logic blocks for this application.

---

## Lesson 19 - Adding Dictionary Items

---

### Objectives

In this lesson, you define a number of dictionary items required for this part of the application. You will mainly be using procedures that have been described earlier in this guide, so you should be familiar with the appropriate developer facilities.

You will also be introduced to validation tables.

Upon completing this lesson you will have:

- Received practice in defining dictionary items
- learned to use validation tables.

---

### Dictionary Requirements

The first task is to create the files used by this system. This involves creating the field specifications, record layouts, and data files. The file and fields you need are shown below:

- An option file with the following fields:
  - option\_key (not necessary for HP ALLBASE/SQL based applications)
  - product\_no
  - option\_no
  - description
  - cost
  - quantity
  - unit\_measure

#### 9-6 Learning Additional Screen Techniques

## Task 1 - Creating Dictionary Field Specifications

The first items you must create are the dictionary field specifications. Most of these field specifications already exist from the previous work you have done, so you won't need to create the *product\_no*, *description*, *cost*, *quantity*, or *unit\_measure* fields.

1. If you are creating the KSAM based or HP TurboIMAGE/iX based application, create both of the specifications described below.
2. If you are creating the HP ALLBASE/SQL based application, only create the first of these specifications.
  - a. **Option Number.** A three digit option specifier. A unique value within any given product.

|                 |                        |
|-----------------|------------------------|
| Field Spec Name | <code>option_no</code> |
| Field Length    | 3                      |
| Edit Code       | N                      |
| Storage Type    | C                      |
| Pad Character   | 0                      |

- b. **Option Key.** A concatenation of the product number and option number forming a unique key to the option data file. You cannot actually specify the concatenation when you define the field specification. Later, you will create some logic that links the product number and option number fields, and sends the output to the *option\_key* field specification.

|                      |                         |
|----------------------|-------------------------|
| Field Spec Name      | <code>option_key</code> |
| Field Length         | 9                       |
| Minimum Entry Length | 9                       |
| Edit Code            | U                       |

When you have created these dictionary field specifications you can create the record layout and the option file.

## Task 2 - Creating the Record Layout and File

If you are developing the KSAM based application, continue reading below.

If you are developing the HP ALLBASE/SQL based application, turn to the instructions for creating ALLBASE/SQL based record layouts.

If you are developing the HP TurboIMAGE/iX based application, turn to the instructions for HP TurboIMAGE/iX Based Applications.

### To create KSAM Based record layout:

1. Create the following record layout.
2. Remember to generate the layout once you have defined it.
  - a. **Option record.** This record contains the seven fields needed to store and retrieve each option item.

#### Record Layout Name. option

| Field | Field Spec.Name | Key # | Duplicates |
|-------|-----------------|-------|------------|
| 1     | option_key      | 1     | N          |
| 2     | product_no      | 2     | Y          |
| 3     | option_no       | 3     | Y          |
| 4     | description     |       |            |
| 5     | cost            |       |            |
| 6     | quantity        |       |            |
| 7     | unit_measure    |       |            |

You can now create the *option* data file.

#### b. Option File.

The option file contains all the option records. Create the file definition, using the details below, and then use the file creation screen to create the physical file.

|                |        |
|----------------|--------|
| File Name      | option |
| File Type      | I      |
| External Name  | option |
| Default Record | 1      |
| Record Layout  | option |

## 9-8 Learning Additional Screen Techniques

You have now completed all the records and files required for this application. Turn to Variables to continue developing your application.

**To create HP ALLBASE/SQL based record layouts:**

1. Create the following record layout.
2. Remember to generate the layout once you have defined it.
  - a. **Option record.** This record contains the six fields needed to store and retrieve each option item.

**Record Layout Name.** option

| Field | Field Spec.Name | Key # | Duplicates |
|-------|-----------------|-------|------------|
| 1     | product_no      |       |            |
| 2     | option_no       |       |            |
| 3     | description     |       |            |
| 4     | cost            |       |            |
| 5     | quantity        |       |            |
| 6     | unit_measure    |       |            |

You can now create the *option* table.

b. **Option Table.**

The option table contains all the option records.

Create the table definition according to the following definition and then use the file creation screen to create the physical table.

|                  |        |
|------------------|--------|
| File Name        | option |
| File Type        | S      |
| External Name    | OPTION |
| SQL DBfileset    | OPTFS  |
| SQL Access Class | 1      |
| Record Layout    | option |

You have now completed all the records and files required for this application. Turn to Variables, to continue developing your application.

**To create the HP TurboIMAGE/iX based record layout:**

For the HP TurboIMAGE/iX based application, you will define the *option* data set, which contains all the details about each option for a product.

The HP TurboIMAGE/iX database *traindb* created by your HP HP ALLBASE/4GL or HP TurboIMAGE/iX administrator also contains automatic master data sets which act as indexes to the *option\_key* and *option\_no* key field specifications. You won't need to define these automatic master data sets in HP ALLBASE/4GL as you do not use them directly in HP ALLBASE/4GL logic. The HP TurboIMAGE/iX database also specifies that the *product\_no* field is linked to the manual master data set *product*.

1. Create the following record layout.
2. Remember to generate the layout once you have defined it.
  - a. **Option record.** This record contains the seven fields needed to store and retrieve each option item.

**Record Layout Name. option**

| Field | Field Spec.Name | Key # | Duplicates |
|-------|-----------------|-------|------------|
| 1     | option_key      | 1     | N          |
| 2     | product_no      | 2     | Y          |
| 3     | option_no       | 3     | Y          |
| 4     | description     |       |            |
| 5     | cost            |       |            |
| 6     | quantity        |       |            |
| 7     | unit_measure    |       |            |

You can now define the *option* data set.

**b. Option File.**

The option file contains all the option records. Create the file definition now.

File Name            **option**  
File Type            **D**  
Database Name        **traindb**  
Schema Data Set     **OPTION**  
Name

**9-10 Learning Additional Screen Techniques**

|                |        |
|----------------|--------|
| Default Record | 1      |
| Record Layout  | option |

You have now completed all the records and files required for this application.

### Task 3 - Creating Variables

This part of the application requires three additional variables for KSAM based and HP TurboIMAGE/iX based applications, and four additional variables for HP ALLBASE/SQL based applications. The variables are described below.

1. Create each variable now.

- a. **confirm.** This variable is used to receive confirmation from the user for a deletion request. The *Q* edit code is a special *Query* code which allows the entry of *Y*, *y*, *N*, or *n* only.

|           |         |
|-----------|---------|
| Name      | confirm |
| Length    | 1       |
| Edit Code | Q       |

- b. **current\_record.** Indicates the current record type, which is either product or option.

|           |                |
|-----------|----------------|
| Name      | current_record |
| Length    | 3              |
| Edit Code | X              |

- c. **option\_status.** Indicates whether the current option record exists.

|           |               |
|-----------|---------------|
| Name      | option_status |
| Length    | 1             |
| Edit Code | X             |

If you are developing the HP ALLBASE/SQL based application, create the following variable. If you are *not* developing the HP ALLBASE/SQL based application, continue reading from the Alphanumeric Constants section below.

- d. **option\_no.** Contains the *option\_no* currently displayed on the *option* window.

|        |           |
|--------|-----------|
| Name   | option_no |
| Length | 3         |

### Learning Additional Screen Techniques 9-11

Edit Code                    N

#### **Task 4 - Creating Alphanumeric Constants**

Two alphanumeric constants are required for the application. They are described below.

1. Create both of these constants now.
  - a. **option.** An indicator of the current record.

|       |        |
|-------|--------|
| Name  | option |
| Value | opt    |
  - b. **product.** An indicator of the current record.

|       |         |
|-------|---------|
| Name  | product |
| Value | prd     |

#### **Task 5 - Creating Numeric Constants**

For KSAM based applications, this application requires two more additional numeric constants.

For HP ALLBASE/SQL based applications, only the *zero* numeric constant is required.

For HP TurboIMAGE/iX based applications, this application requires three more additional numeric constants.

1. Create the constants necessary for your application.
  - a. **All Applications**

|       |      |
|-------|------|
| Name  | zero |
| Value | 000  |

This numeric constant is the value of the first option number for each product. The constant is used to test for the first option.
  - b. **KSAM Based and HP TurboIMAGE/iX Based Applications**

**Numeric Constant - end\_of\_file.**

|       |             |
|-------|-------------|
| Name  | end_of_file |
| Value | 19110       |

#### **9-12 Learning Additional Screen Techniques**



The value of this numeric constant is the file error value returned by the KSAM data manager if a FILE \*NEXT command encounters the end or beginning of a file. In this application it is used to test the file return status value.

c. **HP TurboIMAGE/iX Based Applications**

|       |              |
|-------|--------------|
| Name  | end_of_chain |
| Value | 19115        |

This numeric constant is the file error value returned by the HP TurboIMAGE/iX data manager if a FILE \*NEXT command encounters the end or beginning of a chain of linked detail data set records. The constant is used to test the file return status value.

You have now completed defining all of the constants and variables required by the application.

## **Task 6 - Creating Validation Tables**

The last dictionary item needed to support the field specifications and files is the *unit* validation table. You haven't used validation tables yet so it's described in full.

### **Menu Path**

#### **To access the Validation Tables Screen**

1. From the main menu, select the *Dictionary* option.
2. Select the *Validation Items* option.
3. Choose *Tables*.

#### **Screen Description**

A validation table is used in a similar manner to a validation range. A table contains a number of discrete values that are acceptable values in a field.

A validation table is a table containing up to 51 values.

When you associate a validation table with a screen field, HP ALLBASE/4GL automatically tests the data entered by the user against the values in the table

when data entry is complete. If the value entered exists in the table, processing continues normally.

If the entered value does not exist in the table, HP ALLBASE/4GL displays the message associated with the table. Processing then continues in accordance with the type of message. If you don't define a message in association with the table, HP ALLBASE/4GL displays a system defined ERROR type message by default.

**Note**



In some cases you may need to validate input data against more than 51 values. You can use the CHECK logic command in an after function on the field to validate the entered data against a number of tables. Refer to the *HP HP ALLBASE/4GL Developer Reference Manual* for further information on the CHECK command.

| Developer        |      | Validation Tables |            | tables          |
|------------------|------|-------------------|------------|-----------------|
| Table Name       | unit | Message Name      | unit_error | Secured N (Y/N) |
| (1) Table Values | (18) | (35)              |            |                 |
| g                |      |                   |            |                 |
| kg               |      |                   |            |                 |
| m                |      |                   |            |                 |
| m2               |      |                   |            |                 |
| m3               |      |                   |            |                 |
| in               |      |                   |            |                 |
| ft               |      |                   |            |                 |
| yd               |      |                   |            |                 |
| ml               |      |                   |            |                 |
| l                |      |                   |            |                 |
| ton              |      |                   |            |                 |
| gal              |      |                   |            |                 |
| oz               |      |                   |            |                 |
| lb               |      |                   |            |                 |
| cb               |      |                   |            |                 |
| unit             |      |                   |            |                 |
| (17)             | (34) | (51)              |            |                 |

|              |        |                 |             |             |      |               |
|--------------|--------|-----------------|-------------|-------------|------|---------------|
| Field Specs. | Ranges | Messages 22* 58 | System Keys | Commit Data | Help | Previous Menu |
|--------------|--------|-----------------|-------------|-------------|------|---------------|

**Validation Tables Screen**

To enter the field values:

**9-14 Learning Additional Screen Techniques**

| Field               | Entry                              | Explanation                                                                                                                                                          |             |
|---------------------|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Table Name</b>   | <b>unit</b>                        | This is the name of the table within the application.                                                                                                                |             |
| <b>Secured</b>      | Accept the default.                | This designates whether this item is secured against an unauthorized developer changing the entry.                                                                   |             |
| <b>Message Name</b> | <b>unit_error</b>                  | This is the name of the message that is displayed if the value entered by the user does not match any of the values in the table. You will define this message soon. |             |
| <b>Table Values</b> | Enter the following abbreviations. | The meanings of the abbreviations are listed only for information.                                                                                                   |             |
| <b>g</b>            | gram                               | <b>ml</b>                                                                                                                                                            | milliliter  |
| <b>kg</b>           | kilogram                           | <b>l</b>                                                                                                                                                             | liter       |
| <b>m</b>            | meter                              | <b>ton</b>                                                                                                                                                           | ton         |
| <b>m2</b>           | square meter                       | <b>gal</b>                                                                                                                                                           | gallon      |
| <b>m3</b>           | cubic meter                        | <b>oz</b>                                                                                                                                                            | ounce       |
| <b>in</b>           | inch                               | <b>lb</b>                                                                                                                                                            | pound       |
| <b>ft</b>           | foot                               | <b>cb</b>                                                                                                                                                            | cubit       |
| <b>yd</b>           | yard                               | <b>unit</b>                                                                                                                                                          | single unit |

You can add some other values if you wish.

1. Press the **Commit Data** function key to create the table.
2. When you have committed the table, include it in the field specification by performing the following steps.

| Action                               | Explanation                                                         |
|--------------------------------------|---------------------------------------------------------------------|
| Press <b>Field Specs.</b>            | This displays the field specification screen.                       |
| Enter <b>unit_measure</b>            | This is the name of the field specification that has to be altered. |
| <b>Tab</b> to the <i>Table</i> field | This is the field where you enter the name of the validation table. |
| Enter <b>unit</b>                    | This is the name of the validation table.                           |

## Learning Additional Screen Techniques 9-15

Press **Commit Data** This creates the new field specification.

3. Now go to the screen field details screen and call up the *mb\_order* screen (which was created by the module builder).
4. Call up field number 6 and you will see that the new detail for the field specification has been included automatically.
5. Press the **Generate Screen** function key to incorporate the new table for the field specification in the application.

### **Message - unit\_error**

The last thing you need to do is to create the *unit\_error* message.

1. Go to the message definition screen.
2. If you haven't defined any other messages, tables, or ranges since you defined the validation table, the message name should default to *unit\_error*. If it doesn't, enter the correct name.
3. Assign the message an **ERROR** type code and enter a message to inform the user that the unit of measure just entered is invalid.
4. Then commit the message to generate it.

All of the additional dictionary items for this part of the application are complete.

---

## **Summary**

In this lesson you defined a number of dictionary items for this application.

Among the dictionary items, you defined a validation table.

Validation tables are implemented in much the same manner as validation ranges. The major difference is that a table contains a number of discrete values while a range specifies the lower and upper values to be applied to the field during validation.

When you associate a validation table with a screen field, HP ALLBASE/4GL automatically validates the user's input against the table when data entry

### **9-16 Learning Additional Screen Techniques**

is complete. If the value entered exists in the table, processing continues normally. If the entered value does not exist in the table, HP ALLBASE/4GL displays the message that you have associated with the table. Processing then continues in accordance with the type of the message. If you don't define a message in association with the table, HP ALLBASE/4GL displays a system defined ERROR type message by default.

---

## **Lesson 20 - More Screen Techniques**

---

### **Objectives**

In this lesson you will create a number of screens for the application. The screens you create will include a scroll area, and you will also create some window screens.

This lesson also introduces the screen painter facilities that allow you to use an existing screen as a template for a new screen, and to include line drawing characters on your screens.

Upon completion of this lesson, you will have learned how to:

- Create a scroll area
- Create window screens
- Use a screen template
- Include line drawing characters on screens

---

## Screens for the Enhanced Training Application

This application requires three new windows, and alterations to the *product\_scrn* data screen.

From the main menu, the user selects one of four possible file update actions. These actions are functions that set the appropriate update mode. All four functions then call the same process. The process displays the *product\_scrn* data screen.

The *product\_scrn* data screen contains only fields that are relevant to the product records. A function on the first field of this screen displays a window with headings for a scroll area as well as some line drawing characters which draw a box on the product screen.

When the user presses the `Option` function key, an *option* window is displayed. This window contains the same scroll area headings as the data screen. It also contains the fields relevant to the option record as well as its own arrangement of line drawing characters to complete the box on the product screen as well as providing a box around its own fields.

The exact line and column positioning of the items you create is not critical. Although you do have some freedom as far as the position of the items is concerned, you must keep the fields in the same order as shown on the example screens.

### Task 1 - Altering the Product Data Screen

The *product\_scrn* data screen already contains all of the fields necessary to maintain product records. Now it needs an area for the option window, and an area for scrolling details for the options for an existing product.

First alter the screen header.

1. Enter the following values, and accept the defaults for the other fields.

|                      |                           |
|----------------------|---------------------------|
| Name                 | <code>product_scrn</code> |
| Scroll: Top Line     | <code>18</code>           |
| Scroll: Bottom Line  | <code>22</code>           |
| Scroll: Direction    | <code>D</code>            |
| Window Starting Line | <code>9</code>            |

### 9-18 Learning Additional Screen Techniques

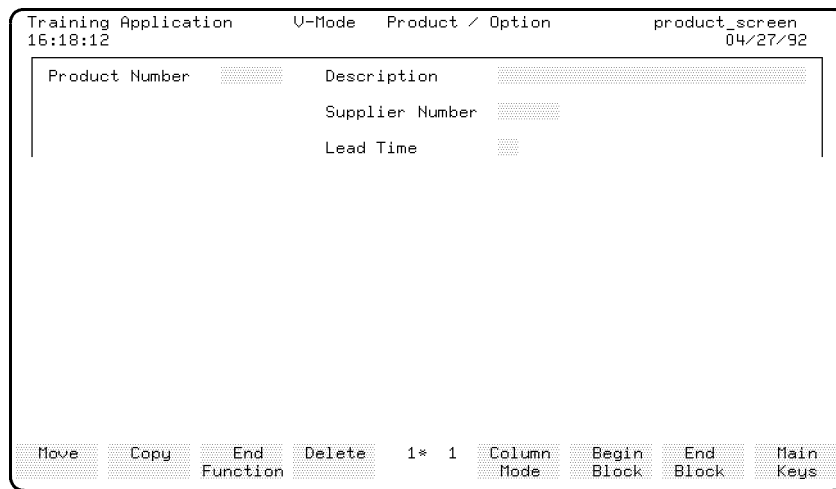
This header specifies a scroll area from line 18 to 22. Whenever a new line of data is displayed in the area with a SCROLL command, the lines already displayed move down as the new line is displayed at the top of the area.

The header also specifies the starting line for a window. Whenever a window is displayed on this screen, the current contents of the screen from line 9 downwards are cleared and the window contents are displayed from line 9 to the bottom of the screen.

- When you have redefined the screen header, press the **Commit Data** function key.

Now you are ready to paint the screen.

- Call up the screen painter and repaint the *product\_scrn* data screen as described below.



The two lines at the top of the screen are similar to those you painted earlier for this screen, except for the V-Mode entry in the center of the first line.

Note the entry *V-Mode*, just before the literal *Product/Option* on the above screen. This is a system item referring to the variable *mode*. Since the same screen is used for adding, modifying, deleting, and reviewing data, this screen item displays the current update mode next to the screen title text.

## Learning Additional Screen Techniques 9-19

**To paint the screen:**

1. Change the first line of the screen to contain the new title.

The maximum length of the *mode* variable is 6 characters, so make sure you leave a space between it and the literal text *Product/Option*.

2. The next task is to create the frame of line drawing characters that surround the items on this screen.

Position the cursor at line 3 column 2, display the `Main Keys` function key set, and use the following steps to create the box on the screen.

| <b>Action</b>                   | <b>Explanation</b>                                                                                                                                                                                          |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Press <code>Special Text</code> | This displays the special text function keys.<br>An asterisk appears in the function key label corresponding to the enhancement modes that are currently <i>on</i> . The default is line drawing mode only. |
| Press <code>Return</code>       | The painter enters text input mode.                                                                                                                                                                         |
| Press <code>Q</code>            | The first character in the first column of the figure below is echoed to the screen. (Refer to your terminal reference manual for a full mapping of line drawing characters to the keyboard characters.)    |
| Press <code>]</code> 76 times   | This extends the line drawing characters to the right giving a line above the prompts and data fields.                                                                                                      |
| Press <code>W</code>            | The first character in the second column of the figure below is echoed to the screen.                                                                                                                       |
| Press <code>Return</code>       | This finishes the special text item creation.                                                                                                                                                               |

**9-20 Learning Additional Screen Techniques**



Here are the line drawing characters you will use in this lesson and their equivalent keyboard characters.

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| r | = | Q | 1 | = | W |
| L | = | A | J | = | S |
| f | = | 1 | f | = | 2 |
| - | = | : |   | = | : |

#### Line Drawing Characters

3. Create the vertical lines on each side of the box, down to, and including, line 8. Each line is made from single character items. You can use the copy facility to speed up the process of creating the lines.
4. Now move the input fields and literal text as shown on the screen earlier. You can use block mode to move the items more efficiently.

The product number literal text should begin at line 4, column 5. The description text should begin at line 4, column 31, and the supplier and lead time text headings should be on lines 6 and 8, respectively, column 31.

5. When you have finished painting the screen, exit from the screen painter to save the screen.

#### To alter screen field details:

The next step is to alter the attributes of the *product\_no* field on the *product\_scrn* screen, using the screen field details screen.

1. Alter the *Prior/After/Both* field to B. This means that the *product\_key\_read* function will be called prior to, and after, entry into the field.
2. This screen is now complete. Commit the screen field details, generate the screen, and return to the screen header screen to create the *option* window.

## Task 2 - Creating the Option Window

The option window screen contains all the fields necessary to maintain records for the options associated with particular products. It is overlaid on the product data screen when the user presses the **Option** function key.

1. First, define the screen header. These are the values you must enter. Accept the defaults for the other values.

```
Screen          option
Screen Type     W
Function Keys   option_keys
```

When HP ALLBASE/4GL displays a window on a data screen, the function keys defined for the window are also displayed. If you have not defined a set of function keys for the window, the function keys from the original screen are not changed.

2. When you have created the screen header, commit the details and then use the screen painter to create the *option* window.

| Option # | Description | Cost | On Hand |
|----------|-------------|------|---------|
|----------|-------------|------|---------|

Move Copy End Function Delete 1\* 1 Column Mode Begin Block End Block Main Keys

**Option Window**

### To paint the screen:

Notice that the *option* window is painted at the very top of the painter screen. When it is displayed, the window is overlaid starting from the window start

## 9-22 Learning Additional Screen Techniques

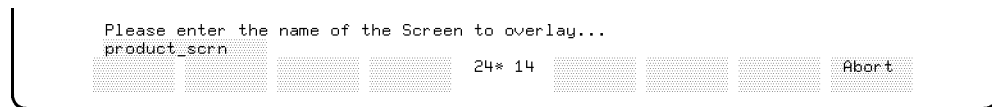
line of the underlying data screen. This means that the line at the top of this screen is aligned with the bottom of the open box on the underlying data screen.

To simplify the task of aligning the fields on the window, there is a screen painter facility that you haven't used yet. This is the *Overlay Screen* capability.

1. With just the blank screen painter screen displayed, follow these steps.

| <b>Action</b>               | <b>Explanation</b>                            |
|-----------------------------|-----------------------------------------------|
| Press <b>More Keys</b>      | This displays a further set of function keys. |
| Press <b>Overlay Screen</b> | A prompt for the overlay screen name appears. |

This is the overlay prompt from the screen painter screen.



### Overlay Prompt

2. Using the overlay prompt, complete these steps:

| <b>Action</b>                                 | <b>Explanation</b>                                                                                                       |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Enter <b>product_scrn</b>                     | HP ALLBASE/4GL prompts you to move the cursor to the starting position of the overlay.                                   |
| Press <b>cursor home</b> , then <b>Return</b> | This moves the cursor to the top left hand corner of the screen. This where you want the overlaid screen to be inserted. |
| Press <b>Return</b>                           | The <i>product_scrn</i> data screen is overlaid onto this screen.                                                        |

You can now paint the window with the new items in correct alignment with the existing items on the product screen, as described below.

**To paint the window:**

1. First delete the unwanted header fields from the top of the *product* screen and use the block move facility to move the rest of the window fields up two lines so that they begin at the top of the screen. The last fields should be on line 6.
2. Copy the horizontal special text item at line 1 to line 7. Then highlight the text item on line 1. With the cursor located first at the left end of the item at line, press **(1)**.
3. Then move to the right end of the item, and press **(2)**. The correct characters are displayed.
4. To correct the line drawing characters at the ends of the item at line 7, use the keys **(A)** and **(S)**.
5. Delete the current prompts and input fields and complete the screen with all of the prompts and input items for the *option* file.

The text literals are shown on the screen image above. The input items are the field specifications in the option file. The last line of text on the window is a heading for the scroll area. This text should be on line 9 of the screen.

6. When you have completed the screen, leave the screen painter so you can define the screen field details for this screen, which are detailed below.

**To define screen field details:**

1. Define the attributes for each of the screen fields listed below. Only those values requiring a specific entry are shown here.
2. Accept the defaults for the other fields.

a. **Field 1**

|                       |                    |
|-----------------------|--------------------|
| Field Spec.Name       | option_no          |
| Primary Data Movement | F-option_no.option |
| Function              | option_key_read    |
| Prior/After/Both      | A                  |

b. **Field 2**

|                       |                      |
|-----------------------|----------------------|
| Field Spec.Name       | description          |
| Primary Data Movement | F-description.option |

**9-24 Learning Additional Screen Techniques**

c. **Field 3**

|                       |               |
|-----------------------|---------------|
| Field Spec. Name      | cost          |
| Primary Data Movement | F-cost.option |

d. **Field 4**

|                       |                   |
|-----------------------|-------------------|
| Field Spec. Name      | quantity          |
| Primary Data Movement | F-quantity.option |
| Required Field        | N                 |

e. **Field 5**

|                       |                       |
|-----------------------|-----------------------|
| Field Spec. Name      | unit_measure          |
| Primary Data Movement | F-unit_measure.option |

3. When you have completed the *option* window, remember to generate it. You can then go on to create the *new\_option* window.

### Task 3 - Adding a new Option Window

This application requires a further window to allow the user to add option number 000 for a new product. This window is almost identical to the *option* window, except for the *option\_no* field. On the *new\_option* window, the *option\_no* field is a display only field.

#### To add the new window:

1. Use the utilities menu copying screen to create a copy of the *option* window called *new\_option*.
2. Use the screen field details screen to modify the details for field number 1 on the *new\_option* window to read as shown below.

**Field 1**

|                       |           |
|-----------------------|-----------|
| Default Data Movement | N-zero    |
| Input/Display         | D         |
| Function Name         | option_no |
| Prior/After/Both      | P         |

3. Generate the screen when you have made these changes. You can now return to the screen header to create the *prod\_wnd* window.

## Task 4 - Creating the The prod\_wnd Window

The last screen you need to create is the product window. This window is displayed when the *option* window is not required. In effect it clears the option window from the screen.

### To create the prod\_wnd window:

1. Define the screen header with the following details.

|               |              |
|---------------|--------------|
| Name          | prod_wnd     |
| Screen Type   | W            |
| Function Keys | product_keys |



| Option # | Description | Cost | On Hand |
|----------|-------------|------|---------|
|----------|-------------|------|---------|

Input Field   Output Field   System Item   Special Text   3\* 1   Layout Keys   Sequence Numbers   More Keys   Exit

**prod-wnd Window**

2. Next, paint the window.

This window has no data fields on it. The easiest way to create it is to overlay the *option* window that you have just created, and then delete all the fields relating to any unwanted option details. Then you can modify the existing line drawing characters to end up with the desired screen image.

3. When you have completed this screen, exit from the screen painter and generate the *prod\_wnd* window. Now that you have created all of the

## 9-26 Learning Additional Screen Techniques

screens required by the application, you can define the function key sets used by the screens.

---

## Summary

In this lesson you created and modified a number of new screens for the application. These screens include a data screen, and three windows. The data screen uses a scroll area, and also has a defined area for displaying the windows.

You also used some new screen painter techniques. These included:

- The overlay screen feature that allows you to use an existing screen as a template for a new screen.
- The special text feature that allows you to include line drawing characters on screens. The special text feature also allows you to specify display enhancements such as color and inverse video for particular screen items.

---

## Lesson 21 - Adding Function Keys

---

### Objectives

When you have completed this lesson, you will have learned how to:

- Define function keys.
- Create the functions that these function keys call.

---

### Function Keys

A function key set can contain up to eight function keys. Each key can execute an item when the user presses the key. Some of the items that can be called from a function key are:

- A function.
- A process.
- A report.
- A screen.
- A help screen.
- Another function key set.

Any type of screen can use a function key set. A function key set is defined separately from the screen, so you can use one function key set with a number of screens. The function key set associated with a screen is named on the screen header screen.

When you define a function key set you can define up to three items for each function key. These items are an action, the text for the function key label, and optionally, a user switch to be assigned to the function key. The status of the user switch is displayed in the function key label.

### 9-28 Learning Additional Screen Techniques



---

## Defining Function Keys

The keys you will define in this lesson are two sets of terminal function keys: the *product\_keys* function keys, which are attached to the *prod\_wnd* window, and the *option\_keys* function keys, which are attached to the *option* and *new\_option* windows.

These sets of keys both display a function key that allows the user to scroll the options for the current product.

The *product\_keys* key set also has a key that allows the user to select option updating. The *option\_keys* function key set has a key that allows the user to return to product updating.

Each function key calls a function to perform the action.

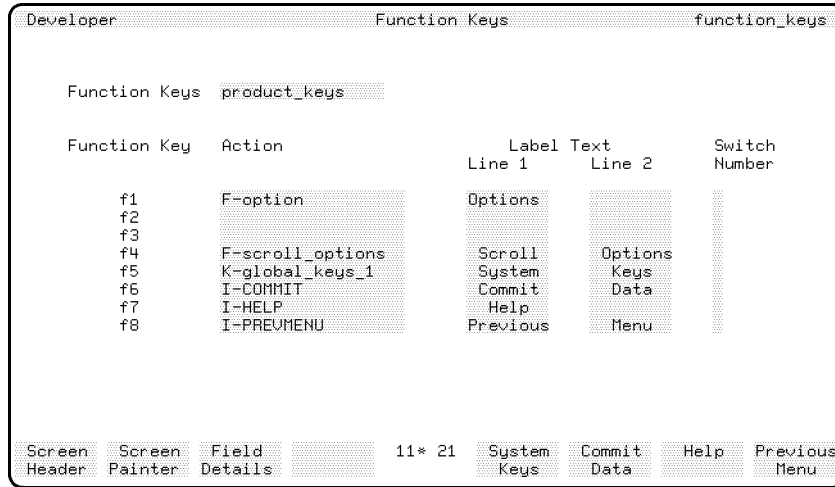
### Menu Path

#### To access the Function Key Screen:

1. From the main menu, select the *Screens* option.
2. Choose *Function Keys*.

### Screen Description

This screen allows you to define the actions, labels, and associated switches for a set of function keys.



### Function Keys Screen

To enter the field values:

| Field         | Entry        | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Function Keys | product_keys | <p>This is the name of the function key set within the application.</p> <p>When you press <b>(Return)</b>, HP ALLBASE/4GL supplies a default set of function key definitions. These default definitions satisfy the basic needs of all screens.</p> <p>You can create your own default set of keys by defining a function key set called <i>default_keys</i> in your application. Refer to the <i>HP ALLBASE/4GL Developer Reference Manual</i> for more information.</p> |

The rest of the screen is divided into eight rows, each containing four fields. There is one row for each key. The function key number is displayed to the left of these fields. The following steps show you how to define function key number 1.

### 9-30 Learning Additional Screen Techniques

**To define function key f1:**

| <b>Field</b>             | <b>Entry</b>            | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Action</b>            | F-option                | <p>This is the action that HP ALLBASE/4GL performs when the user presses function key <b>(f1)</b>. The first character is the action type. In this case <i>F</i> indicates a function. When the user presses the key, the function <i>option</i> is executed.</p> <p>The <i>HP ALLBASE/4GL Developer Reference Manual</i> contains a list of the actions that can be called from function keys, and the prefixes for these actions.</p>                                                                                                                                                                                         |
| <b>Label Text Line 1</b> | Options                 | <p>This is the first line of the function key label that appears on the screen. Try to center the word in the field on the screen.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Label Text Line 2</b> | Leave this field blank. | <p>This is the second line of the function key label that appears on the screen.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Switch Number</b>     | Leave this field blank. | <p>This field is used to associate a user switch with a function key. If you enter a number in this field, an asterisk appears at the end of the first line of the label text on the screen.</p> <p>When the function key set is displayed on an application screen, the asterisk shows the status of the switch. When user switch 1 is on, the asterisk appears. When the switch is off, the asterisk is cleared.</p> <p>Typically you would use this feature to show the end user the status of a function. For example, you could use a function associated with the function key to toggle a particular operating mode.</p> |

**To complete the screen:**

1. Define the next function key as shown below.

**Function Key f4**

|                   |                  |
|-------------------|------------------|
| Action            | F-scroll_options |
| Label Text Line 1 | Scroll           |
| Label Text Line 2 | Options          |

2. Leave the default definitions for the remaining keys.
3. Press **Commit Data** to create this function key set.

**Defining a Second Function Key Set**

1. Now create the *option\_keys* function key set in a similar way, using the following information.

**Function Key Set Name** option\_keys

a. **Function Key 1**

|                   |           |
|-------------------|-----------|
| Action            | F-product |
| Label Text Line 1 | Products  |

b. **Function Key f4**

|                   |                  |
|-------------------|------------------|
| Action            | F-scroll_options |
| Label Text Line 1 | Scroll           |
| Label Text Line 2 | Options          |

2. Leave the default definitions for the remaining keys.
3. Press **Commit Data** to create this function key set.

**9-32 Learning Additional Screen Techniques**

---

## Defining Function Key Logic

Two functions called from the function keys allow the user to switch between updating products and updating options. These functions are called by the *product\_keys* and *option\_keys* function keys.

The logic and description of each function is provided below.

### To create the option function:

This function allows the user to move to the *option* window when a valid product record has been retrieved.

1. Create the *option* function, using the logic commands shown below.
2. Generate the function.

```
1 IF V-product_status <> C-record THEN MESSAGE  
no_product; EXIT  
2 IF V-mode = C-review THEN MESSAGE no_option_review; EXIT  
3 MOVE C-option V-current_record  
4 MOVE C-no_record V-option_status  
5 FILE *BUFFER option  
6 WINDOW option  
7 MOVE "5" *FIELDNO  
8 EXIT
```

The function operates as described below.

- |        |                                                                                                                                                                                                                      |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | Since options can only be accessed after a product record has been retrieved, this step checks that a product record is current. If there is no current product record, it displays an error message and then exits. |
| Line 2 | In review mode, the options can only be reviewed by using the <i>Scroll Options</i> function key. If the current mode is <i>review</i> , this step displays a warning message, and the function exits.               |
| line 3 | This step starts setting up for the option window. It specifies that the current record type is now an <i>option</i> record.                                                                                         |

- Line 4            An option record has not been accessed yet, so the *option\_status* variable is set to *no\_record*.
- Line 5            This step clears the option file buffer. Data may otherwise remain in the option file buffer from a previous update.
- Line 6            This step displays the *option* window. The current contents of the data screen from line 9 downwards are cleared and the window is overlaid on the screen from that line downwards.
- Line 7            This step changes the value of the current field number. The cursor moves to the option number field, field number 5, on the screen when the function finishes.

---

**Note**



Since the function changes the value of \*FIELDNO, HP ALLBASE/4GL automatically commits the current field when this function finishes. If the user presses the **Option** function key while the cursor is on the *product\_no* field, the *product\_key\_read* function is executed as an after function.

---

- Line 8            The function exits.

**To create message for the function key:**

The *option* function displays the *no\_option\_review* message. In *review* mode, options can only be reviewed by pressing the **Scroll Options** function key.

1. Create this message now.

|          |                                                                   |
|----------|-------------------------------------------------------------------|
| Name     | <code>no_option_review</code>                                     |
| Type     | <code>WARN</code>                                                 |
| Contents | "Press the [Scroll Options] function key"<br>"to review options." |

**To create the product function:**

This function allows the user to return to the product screen.

1. Create the *product* function, using the logic commands shown below.
2. Generate the function.

**9-34 Learning Additional Screen Techniques**

```
1 MOVE C-product V-current_record
2 ON 1
3 MOVE "1" *FIELDNO
4 EXIT
```

This function is very simple. The current record mode is first set to *product*. The initial scroll indicator, user switch 1, is set to *on*. This switch is tested in logic that you will create later, and *on* indicates that no option has been read for the product.

Then the number of the *product number* screen field is moved to \*FIELDNO. The function then exits.

When the cursor moves to the first field of the screen the *product\_key\_read* function on that field is performed in its prior state. When you modify that function in the next chapter, you will see that it clears the *option* window area by displaying the *prod\_wnd* window.

---

## Summary

In this lesson, you defined a set of function keys and the functions that are called by them.

A function key set can contain up to eight function keys which execute an item when the user presses a key. Some of the items that can be called from a function key are:

- A function.
- A process.
- A report.
- A screen.
- A help screen.
- Another function key set.

Each function key can have the following items defined for it:

- An action.

This is the item that is executed when the user presses the key.

- Label text.

This is the text that appears on the function key label on the application screen.

- A switch number.

You can associate a user switch with a function key. The status of the switch is shown by the presence or absence of an asterisk in the function key label.

This lesson also introduced an important feature of the screen processing system. The *product* function and the *option* function are both called from function keys. These functions change the value in the communication area field \*FIELDNO. When a function called from a function key changes the value in \*FIELDNO, HP ALLBASE/4GL executes the function, and then immediately performs the input processing for the current screen field when the function exits.

In this application, pressing the **Option** function key while the cursor is on the *product\_no* field executes the *option* function immediately. HP ALLBASE/4GL then completes the input processing for the *product\_no* field. This processing executes the *product\_key\_read* function to retrieve a product record.

## 9-36 Learning Additional Screen Techniques



---

## Lesson 22 - Scrolling Data on the Screen

---

### Objectives

When you have completed this lesson, you will have learned how to

- Use the SCROLL command to display data on the scroll area of a screen.
- Use multiple record layouts in conjunction with a data file.

---

### Understanding the Scrolling System

The *product\_scrn* data screen has a scroll area on it. This lesson shows you how to scroll data in this area.

The scrolling system for this application operates as follows. The first time the user presses the *Scroll Options* function key, the scroll area is cleared and the first option record for the current product is read and displayed in the scroll area. As the user continues to press the *Scroll Options* function key, subsequent option records for the current product are scrolled onto the screen. To do this, the application must maintain a file pointer to the current scroll record so the next record to be scrolled can be easily obtained.

### Task 1 - Creating Secondary File Record Layouts

The application scrolls records from a file which is also being used for the creation or modification of specific records. This requires the maintenance of a work buffer as well as the buffer being used to read records from the file. For KSAM based applications and HP TurboIMAGE/iX based applications, this is achieved by defining a second record layout for the file.

For HP ALLBASE/SQL based applications, tables do not permit multiple record layouts, so the same scrolling system cannot be used. However, you can use a select list to perform the same function as a secondary record layout for a file or table. Using a select list allows you to open a second cursor to retrieve option records without affecting any currently open cursors.

**Learning Additional Screen Techniques 9-37**

If you are developing the KSAM based or HP TurboIMAGE/iX based application, read on. If you are developing the HP ALLBASE/SQL based application, turn to the section HP ALLBASE/SQL Select Lists.

Using two record layouts for the one data file allows you to use two separate file record buffers. This means the application can read and use two different records from the same data file. Using a second record layout with a file does not make any change to the structure of the records in the physical file. The structure of the records in the physical file is always determined by the structure of the default record layout for the file. Secondary record layouts simply provide a means of reading or writing records through different file record buffers.

The structure of a secondary record layout for a file does not need to be identical to the default record layout. If required, you can obtain a different “view” of a file record by reading it through a secondary record layout defined with different field specifications.

In this application, both record layouts for the *option* file have exactly the same structure.

**To create a secondary record layout:**

1. Use the utilities copying function to copy the existing *option* record layout to a new layout called *option\_scroll*.
2. Remember to generate the record layout after you have copied it.
3. Modify the existing *option* file definition by adding the new *option\_scroll* record layout to the list of record layouts as the second layout, using the file/SQL table definition screen.

The default record layout is still the *option* record layout.

All references to the *option* file so far have named only the file, with no record layout specifier. In this form the default record layout is always used, so the existing code remains unchanged.

The next section describes the system HP ALLBASE/SQL based applications use to maintain multiple buffers. Skip this section, and turn to “Task 2 - Creating the Scrolling Functions.”

**9-38 Learning Additional Screen Techniques**

## Creating HP ALLBASE/SQL Select Lists

You can think of a select list as being a “virtual” file. A select list consists of columns, just like an HP ALLBASE/SQL table. The columns of a select list can be derived from one or more base tables or views. The values in a select list column can be taken directly from a column in an existing table or view, or can be values derived in any of the following ways:

- Values computed from a base table column using an arithmetic expression.
- Values computed from the values in various base table columns using an arithmetic expression.
- Values computed from a group of base table column values with an aggregate function (AVG, MAX, MIN, SUM, and COUNT).
- Constants, or values computed from an expression involving constants.

### Defining a Select List

Defining a select list is similar to defining many other items. You must define a select list header, then complete the details for the select list, and finally generate the select list.

Each entry in a select list definition has the following format:

*field\_specification* [ = *column\_definition*]

The *field\_specification* must be the name of a dictionary field specification. The term *column\_definition* means any valid SQL column definition. You can use SQL aggregate functions and expressions in the column definitions for SQL select lists. Note that you must terminate each column definition except the last with a comma (,). Each column definition does not necessarily need to be on a separate line of the screen. You can put more than one column definition on the same line.

To abbreviate the entry of select lists, you can omit the right hand side of each entry if the column definition is exactly the same as the field specification name, and no ambiguity results from the omission.

This part of the application uses a select list called *optscrol*. It uses the *option\_no*, *description*, *cost*, *quantity*, and *unit\_measure* fields from the option

table. These fields contain the information necessary to scroll the option details on the screen.

The screens you use to define select lists are accessible from the database items menu in the dictionary menu.

**To access the Select List Screen:**

1. From the main menu, select the *Dictionary* option.
2. Select the *Database Items* option.
3. Choose *SQL Select List Header*.

This screen is similar in appearance and operation to the other header screens.

**To define a select list:**

1. Go to the *Select List Header* screen and define a header for the *optscrol* select list.
2. When you have defined the select list header, go to the *Select List Details* screen to define the details for the select list, as shown below.

This screen contains a free-format data entry area that allows you to enter the details for a select list.

3. Define the select list as follows:

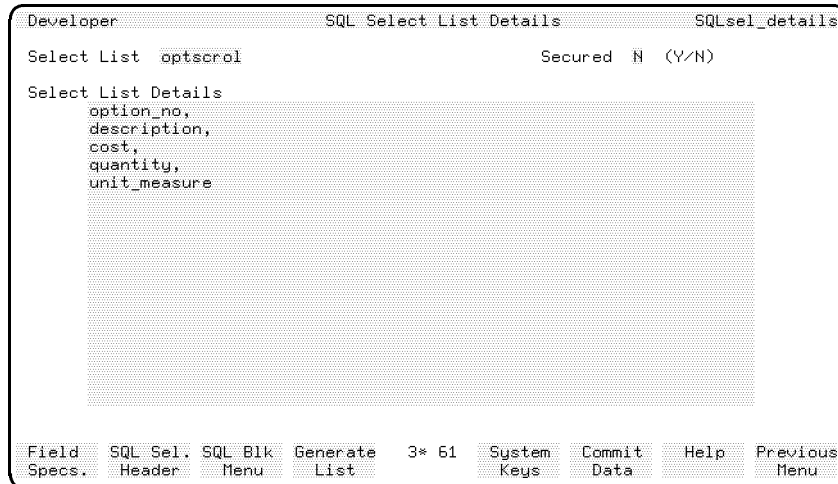
```
option_no,  
description,  
cost,  
quantity,  
unit_measure
```

This select list is equivalent to the following expanded expression.

```
option_no      = option_no,  
description    = description,  
cost           = cost,  
quantity       = quantity,  
unit_measure   = unit_measure
```

**9-40 Learning Additional Screen Techniques**

In this example, the column definitions for the select list are exactly the same as the field specification names, so you can omit the right hand side of each item.



### SQL Select List Details

#### Generating a Select List

You must generate a select list to convert the definition to an executable run-time format.

#### To generate the select list:

1. When you have completed the select list details, press the **Commit Data** function key.
2. Then press the **Generate List** function key to generate the select list.

During generation, HP ALLBASE/4GL uses the field specification names on the left hand side of each select list column definition to create and generate a record layout for the select list.

Since generating a select list builds a record layout, a select list cannot have the same name as an existing record layout. It also cannot have the same name as an existing file or base table.

### Learning Additional Screen Techniques 9-41

## Task 2 - Creating The Scrolling Functions

You can now create the functions that scroll data on the screen. You do this by defining the logic commands for the function.

### Function - *scroll\_options*

Pressing the **Scroll Options** function key executes the *scroll\_options* function. This function calls two further functions to retrieve the first and subsequent option records for the current product.

The function is almost the same for all three data managers. The only difference is the first line, which defines the name of the file.

### To define the scrolling function logic:

1. To define the first line of logic for KSAM based and HP TurboIMAGE/iX based applications, enter the following:

```
1 DEFINE %FILE% option.option_scroll
```

**or**

2. To define the first line of logic for HP ALLBASE/SQL based applications, enter the following:

```
1 DEFINE %FILE% optscroll
```

Independent of the data manager you are using, the rest of the function is as shown below.

3. To create the rest of the function, enter the lines below:

```
1 ...
2 IF V-product_status <> C-record THEN
MESSAGE no_product; EXIT
3 IF 1 *ON THEN VISIT get_1st_option; ENTER 7
ELSE VISIT get_next_option
4 IF 1 *OFF THEN ENTER 7
5 SCROLL 9 "***** End of Options for Product #"
```

## 9-42 Learning Additional Screen Techniques

```

S-product_no.product_scrn "*****"
6 EXIT
7 SCROLL 6 F-option_no.%FILE% 4 F-description.%FILE%
5 F-cost.%FILE% 6 F-quantity.%FILE% 1 F-unit_measure.%FILE%
8 EXIT

```

The *scroll\_options* function operates as follows.

- Line 1            This step introduces the DEFINE command. The DEFINE command creates an identifier that is expanded whenever it is referenced in the logic block. The identifier in this case is *%FILE%*.
- It is expanded to *option.option\_scroll* every time it is referenced, for KSAM based and HP TurboIMAGE/iX based applications.
- For HP ALLBASE/SQL based applications, it is expanded to *optscrol* every time it is referenced.
- As the file specifier is quite long, using the identifier simplifies the entry of commands. Any DEFINE commands must precede the other commands in a logic block. You can have more than one DEFINE command at the start of a logic block.
- Line 2            This step checks to ensure that a product record is current. If there is no current product record, this step displays a message and the function exits.
- Line 3            An earlier function (for example, *product\_key\_read*) may set user switch number 1 *on*. This step checks the status of the switch. If the switch is *on*, no previous option has been read for this product so it is necessary to retrieve the first option for the product. The function *get\_1st\_option* reads the first option record. When the *get\_1st\_option* function exits, control passes to step 7 of this function. Steps 4 to 6 are not executed if an option has not yet been retrieved.
- If user switch 1 is *off*, an option has already been read for this product and the next one must be retrieved. The function *get\_next\_option* reads the next option record.

A clue to entering this command in the IF window: enter \*ON in the IF Test field, and leave the Data Name 2 field blank.

Line 4 This step is only executed after a previous option has been read for this product. The function *get\_next\_option* sets user switch 1 *on* if no more options for the product are found. If the switch is *off*, a record has been found and control passes to step 7 where the data is scrolled on the screen.

Line 5 This SCROLL command is executed when the last of the options has been found. The SCROLL command takes the specified parameters and concatenates them to form a line of text that is displayed in the scroll area. Any lines already displayed in the scroll area are moved one line in the defined scroll direction and the new line of text is displayed on the first line of the scroll area.

The first SCROLL parameter is 9. This is displayed as a string of nine spaces so the second SCROLL parameter, a text literal, is preceded by nine spaces. The third SCROLL parameter is a screen field and the last SCROLL parameter is a concluding literal. Separate the parameters with spaces when you enter the command.

Line 6 The function exits at this point if all the options have been scrolled.

Line 7 This is the step that actually scrolls the option data on the screen.

This step shows the use of the identifier %FILE% that was defined in step 1. For example, the parameter:

`F-option_no.%FILE%`

is expanded to read:

`F-option_no.option.option_scroll`

for KSAM based applications and HP TurboIMAGE/iX based applications,

**or**

## 9-44 Learning Additional Screen Techniques



F-option\_no.optscrol

for HP ALLBASE/SQL based applications.

In the SCROLL command, each of the file fields is separated by a number of spaces so they are aligned with the column headings painted on the screen.

Line 8           The function exits at this point after scrolling the details of an option record.

### **Creating the Functions Called from the scroll\_options function**

The next step is to create the two functions that are called from the *scroll\_options* function. These functions differ slightly between data managers.

- If you are developing the KSAM based application, continue reading below.
- If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications for details about the *get\_1st\_option* and *get\_next\_option* functions.
- If you are developing the HP TurboIMAGE/iX based application, turn to HP TurboIMAGE/iX Based Applications for details about the *get\_1st\_option* and *get\_next\_option* functions.

### **KSAM Based Applications**

#### **Function - *get\_1st\_option***

The next function that you need to create is the *get\_1st\_option* function. This function is called by the *scroll\_options* function to retrieve the first option record for the current product. Since the processing rules for this application require that each product must have at least one option (number 000), this function should always retrieve an option record. Failure to find a record indicates a serious file problem.

#### **To create the get\_1st\_option function:**

1. Create the function, using the logic commands shown here.
2. When finished, generate the function.

```

1 DEFINE %FILE% option.option_scroll
2 DISPLAY *RESET=S ""
3 FILE *FIND %FILE% *KEY= F-product_no.product ; ENTER 9
4 FILE *NEXT %FILE%
5 IF F-product_no.%FILE% <> F-product_no.product
THEN ENTER 9
6 SCROLL 9 "***** Start of Options for Product #"
S-product_no.product_scrn "*****"
7 OFF 1
8 EXIT
9 MESSAGE prod_corrupt
10 PROCEED main; NOTE Product/option record corrupted
because option 000 does not exist. Clear buffers and display
the main menu so the user can delete the product.

```

The *get\_1st\_option* function operates as follows:

- |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1           | This step uses the <b>DEFINE</b> command to create an identifier that is expanded whenever it is referenced in the logic block.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Line 2           | The <b>DISPLAY</b> command is another command used to manipulate the scroll area on a screen. It displays a line of data on a specific line in the scroll area. The <b>*RESET=S</b> option clears the scroll area. The empty parameter ("" ) specifies that nothing is displayed on the first line of the scroll area.                                                                                                                                                                                                                                                                                                |
| Lines 3, 4 and 5 | <p>These steps search for a record with a key matching or greater than the value specified, retrieve the record, and test to check whether the record retrieved has the correct product number.</p> <p>The <b>FILE *FIND</b> command searches a file for a record with a key greater than or equal to the value specified. As the <i>option_key</i> field in this application is the concatenation of the <i>product_no</i> and the <i>option_no</i> fields, a search with the key equal to the product number finds the first option for that product. If no suitable record is found, control passes to step 9.</p> |

## 9-46 Learning Additional Screen Techniques

The FIND command does not read the record contents into the file buffer. It simply positions a file pointer to the record. You must use a FILE \*NEXT command to read the record into the file record buffer.

Since the FILE \*FIND command searches for a key value equal to or greater than the specified key, it may find a record with a greater product number than the one specified in step 3.

Step 5 checks to see if the record retrieved by step 4 has the correct value in the product number field. If the value in the product number field does not match the current product number, control passes to step 9.

Line 6       The SCROLL command scrolls a heading line in the scroll area.

Line 7       This command sets user switch number 1 to *off*. The calling function (*scroll\_options*) tests the status of this switch when the user next presses the **Scroll Options** function key. If this switch is *off*, the next option is retrieved.

Line 8       The function exits.

Line 9       Control only passes to this step if an option record for the product has not been found. Failure to find an option record means that the record for option 000 does not exist, suggesting that the product or option file is corrupted. This condition is detected if an error occurs in step 3, or the product number in the record retrieved from the file is not the current product number. This step displays a message telling the user that records may be corrupted.

Line 10      This step demonstrates a method for recovering from a potentially serious problem. The PROCEED command in this step immediately terminates any current activity in the application, clears the file and screen buffers, and closes all data files.

The NOTE command allows you to describe individual lines within a logic block. HP ALLBASE/4GL ignores all the text following a NOTE command.

## Learning Additional Screen Techniques 9-47

The process *main* displays the menu *main* to allow the user to delete or correct the corrupted record.

Once you have defined and generated the function, create the message and the process required by the function. These are described below.

**To create the messages:**

1. Make the entries shown for each message.

- a. **Message** - *prod\_corrupt*

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Name     | <b>prod_corrupt</b>                                              |
| Type     | <b>WARN</b>                                                      |
| Contents | "Product record may be corrupted. "<br>"Delete and enter again." |

- b. **Process** - *main*

The *get\_1st\_option* function calls the *main* process if the product record is corrupted. This process clears all file and screen buffers and then displays the *main* menu. The user can then select another operation from the menu to delete the product or correct the problem. The process is a simple one. The SCREEN command calls the *main* menu.

```
1 SCREEN main
```

**Creating the last Function**

**Function** - *get\_next\_option*

The last function to create is the *get\_next\_option* function. This function is called by the *scroll\_options* function to retrieve the second and subsequent options for the current product.

**To create the get\_next\_option function:**

1. Enter the logic commands shown below.

```
1 DEFINE %FILE% option.option_scroll  
2 FILE *READ %FILE% ; ENTER 7
```

**9-48 Learning Additional Screen Techniques**

```

3 FILE *NEXT %FILE% ; ENTER 5
4 IF F-product_no.%FILE% = F-product_no.product THEN EXIT
5 ON 1
6 EXIT
7 ON 1
8 VISIT get_1st_option
9 EXIT

```

2. Generate the function.

This function operates as shown below.

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This defines an abbreviation for the <i>option.option_scroll</i> reference.                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Line 2 | Between the time of the scrolling of the last record and the current scroll request, the user may have read the option file via the default buffer. This will not directly affect the data in the second buffer but it may affect the file pointer. This step relocates the file pointer by reading the file for the contents of the current scroll record buffer. However, this step will fail if the last scrolled record has been deleted. Control passes to step 7 if this situation occurs. |
| Line 3 | This next record is retrieved from the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Line 4 | This step tests the new record to ensure that it refers to the correct product by testing the value in the <i>product_no</i> field. If the retrieved value is correct, the function exits with the next option details in the correct buffer.                                                                                                                                                                                                                                                    |
| Line 5 | This step sets user switch number 1 <i>on</i> if the end of the file is reached in step 3, or the product number changes as determined in step 4. This indicates to the calling function that the last option for the product has been found.                                                                                                                                                                                                                                                    |
| Line 6 | The function exits.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Line 7 | This step and the next step are only executed if the original record is not found again in step 2. This step sets switch number 1 <i>on</i> .                                                                                                                                                                                                                                                                                                                                                    |

Line 8            This step attempts to locate the first option for the product by visiting the function *get\_1st\_option* again. The return from this step is the same as if the calling function visited *get\_1st\_option* directly.

Line 9            The function exits.

These functions complete the processing required for a fail safe scrolling mechanism.

You cannot test any of your screens or functions yet. At the end of the next chapter all the logic for the application will exist, and you can then see the scrolling functions in action.

Turn to the summary of this lesson.

### **Creating HP ALLBASE/SQL Functions**

#### **Function - *get\_1st\_option***

The next function that you need to create is the *get\_1st\_option* function. This function is called by the *scroll\_options* function to retrieve the first option record for the current product. Since the processing rules for this application require that each product must have at least one option (number 000) this function should always retrieve an option record. Failure to find a record indicates a serious file problem.

#### **To create the *get\_1st\_option* function:**

1. Enter the logic commands shown below.

```
1 DEFINE %FILE% optscrol
2 DISPLAY *RESET=S ""
3 SQL get_optscrol
4 FILE *NEXT %FILE% ; ENTER 9
5 IF F-option_no.%FILE% <> N-zero THEN ENTER 9
6 SCROLL 9 "***** Start of Options for Product #
" S-product_no.product_scrn " *****"
7 OFF 1
8 EXIT
```

### **9-50 Learning Additional Screen Techniques**

9 **MESSAGE** prod\_corrupt  
10 **PROCEED** main ; **NOTE** Product/option records corrupted  
since option 000 does not exist. Clear buffers and display  
main menu so user can delete the product.

2. Generate the function.

The *get\_1st\_option* function operates as follows:

- Line 1            This step uses the **DEFINE** command to create an identifier that is expanded whenever it is referenced in the logic block.
- Line 2            The **DISPLAY** command is another command used to manipulate the scroll area on a screen. It displays a line of data on a specific line in the scroll area. The **\*RESET=S** option clears the scroll area. The empty parameter ("" ) specifies that nothing is displayed on the first line of the scroll area.
- Lines 3, 4 and 5    These steps search for a record with a key matching or greater than the value specified, retrieve the record, and test to check whether the record retrieved has the correct product number.
- This function calls the SQL logic block *get\_optscrol*, which you will create soon. This SQL logic block retrieves the options for the current product. This select command declares and opens a cursor on the *optscrol* select list. The active set for this cursor consists of all the option records for the current product. The records are ordered by option number. The **FILE \*NEXT** command positions the cursor on the first record in the active set, and retrieves the record into the select list buffer.
- Step 5 checks to see if the record retrieved by step 4 has the correct value in the option number field. If the value in the option number field does not match '000', the contents of the *zero* numeric constant, (the first record), control passes to step 9.
- Line 6            The **SCROLL** command scrolls a heading line in the scroll area.
- Line 7            This command sets user switch number 1 to *off*. The calling function (*scroll\_options*) tests the status of this switch when

## Learning Additional Screen Techniques 9-51

the user next presses the **Scroll Options** function key. If this switch is *off*, the next option is retrieved.

Line 8

The function exits.

Line 9

Control only passes to this step if an option record for the product has not been found. This condition is detected if an error occurs in step 4, or the option number in the record retrieved from the file does not equal "000" in step 5. Failure to find an option record means that the record for option 000 does not exist, suggesting that the product or option file is corrupted. This step displays a message telling the user that records may be corrupted.

Line 10

Control only passes to this step if the record for option number 000 cannot be found. It demonstrates a method for recovering from a potentially serious problem. The PROCEED command in this step immediately terminates any current activity in the application, clears the file and screen buffers, and closes all data files.

The NOTE command allows you to describe individual lines within a logic block. HP ALLBASE/4GL ignores all the text following a NOTE command.

The process *main* displays the menu *main* to allow the user to delete or correct the corrupted record.

Once you have defined and generated the function, create the SQL logic block, the message, and the process required by the function. These are described below.

#### **To create the get\_optscrol logic block:**

This is the SQL logic block that is called from the *get\_1st\_option* function.

1. Create the logic block using the block details listed below.

```
SELECT :optscrol FROM sqlgrp.option
  WHERE product_no = F-product_no.product
  ORDER BY option_no;
```

## **9-52 Learning Additional Screen Techniques**



2. Generate the logic block.

This SQL logic block demonstrates the use of a select list in a SELECT command. Note that you must precede the select list name with a colon(:). This SELECT command also includes an ORDER BY clause to retrieve the options for a product in option number order.

This select command declares and opens a cursor on the *optscrol* select list. The active set for this cursor consists of all the option records for the current product. The records are ordered by option number. The FILE \*NEXT command in the function *get\_1st\_option* positions the cursor on the first record in the active set, and retrieves the record into the select list buffer.

**To create the prod\_corrupt message:**

The *get\_1st\_option* function requires this message.

1. Create the message now, using the information shown below.

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Name     | prod_corrupt                                                     |
| Type     | WARN                                                             |
| Contents | "Product record may be corrupted. "<br>"Delete and enter again." |

**Process - main**

The *get\_1st\_option* function calls the *main* process if the product record is corrupted. This process clears all file and screen buffers and then displays the *main* menu. The user can then select another operation from the menu to delete the product or correct the problem. The process is a simple one. The SCREEN command calls the *main* menu.

**To create the main process:**

1. Create this process, using the information below.

```
1 SCREEN main
```

2. Generate the process.

**Function - get\_next\_option**

The last function to create is the *get\_next\_option* function. This function is called by the *scroll\_options* function to retrieve the second and subsequent options for the current product.

**To create the *get\_next\_option* function:**

1. Create this function using the logic commands shown here.

```
1 FILE *NEXT optscrol ; ENTER 3
2 EXIT
3 ON 1
4 EXIT
```

2. Generate the function.

This function reads the next record for the active set for the *optscrol* select list. If a record is retrieved successfully the function exits. If a record is not found (indicating an end-of-file condition and no further option records exist) the function sets switch number 1 on, and then exits. When set *on*, switch number 1 resets the scrolling system.

When you modify the *product\_key\_read* function in the next chapter, you will add a step to set switch number 1 *on*. This is necessary to force execution of the function *get\_1st\_option* if the user has pressed the **Scroll Options** function key after pressing the **Commit Data** function key.

Executing the function *get\_1st\_option* declares and opens a cursor on the *optscrol* select list. This is necessary because the COMMIT WORK command called by the *product* process closes all open cursors. Attempting to execute the function *get\_next\_option* without first declaring and opening a cursor generates an error condition.

These functions complete the processing required for a fail safe scrolling mechanism.

You cannot test any of your screens or functions yet. At the end of the next chapter all the logic for the application will exist, and you can then see the scrolling functions in action.

Turn to the summary of this lesson.

**9-54 Learning Additional Screen Techniques**

## Creating HP TurboIMAGE/iX Functions

### Function - *get\_1st\_option*

The next function that you need to create is the *get\_1st\_option* function. This function is called by the *scroll\_options* function to retrieve the first option record for the current product. Since the processing rules for this application require that each product must have at least one option (number 000) this function should always retrieve an option record. Failure to find a record indicates a serious file problem.

### To create the *scroll\_options* function:

1. Use the logic commands shown here to create the function.

```
1 DEFINE %FILE% option.option_scroll
2 DISPLAY *RESET=S ""
3 FILE *FIND %FILE% *INDEX=product_no *KEY= F-product_no.product ; ENTER 8
4 FILE *NEXT %FILE%
5 SCROLL 9 "***** Start of Options for Product #"
S-product_no.product_scrn " *****"
6 OFF 1
7 EXIT
8 MESSAGE prod_corrupt
9 PROCEED main; NOTE Product/option record corrupted
because option 000 does not exist. Clear buffers and display
the main menu so the user can delete the product.
```

2. Generate the function.

The *get\_1st\_option* function operates as follows:

|        |                                                                                                                                                                                                                                              |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This step uses the <b>DEFINE</b> command to create an identifier that is expanded whenever it is referenced in the logic block.                                                                                                              |
| Line 2 | The <b>DISPLAY</b> command is another command used to manipulate the scroll area on a screen. It displays a line of data on a specific line in the scroll area. The <b>*RESET=S</b> option clears the scroll area. The empty parameter ("" ) |

- specifies that nothing is displayed on the first line of the scroll area.
- Lines 3 and 4 These steps search for a record that matches the value specified, and retrieve the record.
- The FILE \*FIND command searches a file for a record with the product key field equal to the value specified. As the *option\_key* field in this application is the concatenation of the *product\_no* and the *option\_no* fields, a search with the key equal to the product number finds the first option for that product. If no suitable record is found, control passes to step 8.
- The FIND command does not read the record contents into the file buffer. It simply positions a file pointer to the record. You must use a FILE \*NEXT command to read the record into the file record buffer.
- Line 5 The SCROLL command scrolls a heading line in the scroll area.
- Line 6 This command sets user switch number 1 to *off*. The calling function (*scroll\_options*) tests the status of this switch when the user next presses the **Scroll Options** function key. If this switch is *off*, the next option is retrieved.
- Line 7 The function exits.
- Line 8 Control only passes to this step if an option record for the product has not been found. Failure to find an option record means that the record for option 000 does not exist, suggesting that the product or option file is corrupted. This condition is detected if an error occurs in step 3, or the product number in the record retrieved from the file is not the current product number. This step displays a message telling the user that records may be corrupted.
- Line 9 This step demonstrates a method for recovering from a potentially serious problem. The PROCEED command in this step immediately terminates any current activity in the application, clears the file and screen buffers, and closes all data files.

## 9-56 Learning Additional Screen Techniques

The NOTE command allows you to describe individual lines within a logic block. HP ALLBASE/4GL ignores all the text following a NOTE command.

The process *main* displays the menu *main* to allow the user to delete or correct the corrupted record.

Once you have defined and generated the function, create the message and the process required by the function. These are described below.

**To create the prod\_corrupt message:**

1. Make the entries shown below.

**Message - *prod\_corrupt***

|          |                                                                  |
|----------|------------------------------------------------------------------|
| Name     | prod_corrupt                                                     |
| Type     | WARN                                                             |
| Contents | "Product record may be corrupted. "<br>"Delete and enter again." |

**Process - *main***

The *get\_1st\_option* function calls the *main* process if the product record is corrupted. This process clears all file and screen buffers and then displays the *main* menu. The user can then select another operation from the menu to delete the product or correct the problem. The process is a simple one. The SCREEN command calls the *main* menu.

**To create the main process:**

1. Make the entries shown here.

1 SCREEN main

2. Generate the process.

**Function - *get\_next\_option***

The last function to create is the *get\_next\_option* function. This function is called by the *scroll\_options* function to retrieve the second and subsequent options for the current product.

**To create the `get_next_option` function:**

1. Make the entries shown below.

```
1 DEFINE %FILE% option.option_scroll
2 FILE *NEXT %FILE% ; ENTER 4
3 IF F-product_no.%FILE% = F-product_no.product THEN EXIT
4 ON 1
5 EXIT
```

2. Generate the function.

This function operates as shown below.

- |        |                                                                                                                                                                                                                                               |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This defines an abbreviation for the <i>option.option_scroll</i> reference.                                                                                                                                                                   |
| Line 2 | This next record is retrieved from the file.                                                                                                                                                                                                  |
| Line 3 | This step tests the new record to ensure that it refers to the correct product by testing the value in the <i>product_no</i> field. If the retrieved value is correct, the function exits with the next option details in the correct buffer. |
| Line 4 | This step sets user switch number 1 <i>on</i> if the end of the file is reached in step 2, or the product number changes as determined in step 3. This indicates to the calling function that the last option for the product has been found. |
| Line 5 | The function exits.                                                                                                                                                                                                                           |

If a file manipulation function is executed, the scrolling functions will start from the first function again.

These functions complete the processing required for a fail safe scrolling mechanism.

You cannot test any of your screens or functions yet. At the end of the next chapter all the logic for the application will exist, and you can then see the scrolling functions in action.

**9-58 Learning Additional Screen Techniques**

---

## Summary

In this lesson you used the SCROLL command to display data in the scroll area of a screen.

For KSAM based and HP TurboIMAGE/iX based applications, this lesson also introduced the FILE \*FIND command. The FILE \*FIND command searches a file for a record with a key equal to or greater than a specified value. You can then use the FILE \*NEXT command to retrieve the record.

You also associated a secondary record layout with a data file. This enables the application to read and use two different records from the same data file.

For HP ALLBASE/SQL based applications, this lesson introduced SQL select lists.

You used a select list to retrieve data from an HP ALLBASE/SQL base table. Using a select list allows you to declare and open an additional cursor without changing any existing cursors.

In this application, the *optscrol* select list allows you to retrieve information from option records for a product, without changing the existing cursor used to retrieve the current option record that may be displayed on the option window.

Select lists have the following characteristics:

- A select list is a “virtual” file. You can reference fields on a select list as though they are file record fields.
- A select list definition consists of the select list header, and the select list details.
- Each entry in a select list has the format:

*field\_spec* [= *column\_definition*],

Columns can be derived directly from columns of existing tables or views, or can be derived using SQL expressions.

To simplify the procedure for defining a select list, you can omit the right hand side of each entry if the column definition is exactly the same as the field specification name.

- A select list must be generated. When HP ALLBASE/4GL generates a select list, it creates a generated record layout using the field specifications on the left hand side of the select list entries.

Before you move onto the next chapter, you will be introduced to user help screens.



---

## Lesson 23 - Creating User Help Screens

---

### Objectives

When you have completed this lesson, you will have learned how to create user help screens.

### Defining Help Screens

HP ALLBASE/4GL allows you to define three levels of help. These are *message help*, *field help*, and *screen help*.

The procedures for defining the different types of help screens are identical. The way these screens are linked to the application determines the way HP ALLBASE/4GL displays them when the user presses the **Help** function key.

#### To access the Help Screens screen:

1. From the main menu, select the *Dictionary* option.
2. Choose *Help Screens*.

#### Description

A help screen is a free format area of 16 lines that can each contain 60 characters. When displayed, it is centered on the screen.



### Help Screens Screen

#### Creating Field Level Help

The help screen you will define first is an example of field level help.

#### To enter the field values:

| Field            | Entry        | Explanation                                                                                                                                                                                                                                                                                                     |
|------------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Help Screen Name | product_no   | This is the name of the help screen. In this case, the help screen is associated with the <i>product_no</i> field.                                                                                                                                                                                              |
| Next Help Screen | Leave blank. | You are not limited to just one help screen. Entering the name of a further help screen in this field links a further help screen to this screen. When the user presses the <b>More Help</b> function key on this help screen, the next help screen named here is displayed. This example only uses one screen. |

#### 9-62 Learning Additional Screen Techniques

| Field                | Entry                                | Explanation                                                                                                                                                   |
|----------------------|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Help Screen Contents | Enter some text for the help screen. | This is a free form area where you can insert lines, delete lines, insert characters and delete characters as required. You can use any printable characters. |

#### To complete and commit the screen:

1. Press the **Commit Data** function key when you have finished creating the help screen.

When you created the *product\_scrn* data screen using *product\_no* field specification, you accepted the default help name of *product\_no* for the field. Next time you run the application and the cursor is on the product number field, press the **Help** function key. The help screen you have just defined will be displayed.

#### Creating Message Level Help

1. Modify the *product\_no\_error* message so it specifies a help screen name of *product\_no\_error*. This message is invoked when the *product\_no* range check fails.
2. Create a help screen called *product\_no\_error*, to tell the user which values are valid for the field.
3. Run the application again and enter an invalid product number in the *product\_no* field on the *product\_scrn* screen. The error message is displayed and the field is error highlighted.
4. Press the **Help** function key. The help message for the error is displayed. Note that the application screens will not work as described in this chapter because the logic to call these screens has not been created yet.

This is an example of a message level help. The same process applies for ABORT and QUERY type messages.

#### Creating Screen Level Help

Now create a help screen called *product\_scrn* to give the user general assistance in using the *product\_scrn* screen.

1. Go to the *product\_scrn* screen header and ensure that the help screen name is *product\_scrn*.
2. Run the application again.
3. Press the **Screen Help** function key after you have called the field help from the product number field. Screen help will then be displayed.

If you want to allow the user to call up screen help directly from a field, you must ensure that its field help name is blank.

---

## Summary

HP ALLBASE/4GL allows you to define help screens associated with messages, screen fields, and screens. If you need to display more information than you can fit on one screen, you can link further help screens to any help screen. Now you may complete the Self Test Questions on the following page.

---

## Self Test Questions

**Question 1.** How many values can a validation table contain? If you wish to check user input against more than the maximum number of values, is there any way to do so?

**Question 2.** How do you specify a scroll area on a screen?

**Question 3.** In The New Option Window, you use a default data movement for a field. What action does the default data movement perform?

**Question 4.** How do you specify where a window will appear over a data screen?

**Question 5.** Name three types of items that can be executed when the user presses a function key?

**Question 6.** What does the SCROLL command do?

**Question 7. HP ALLBASE/SQL Applications Only**

### 9-64 Learning Additional Screen Techniques

What is a select list? Is it a logic item or a dictionary item?

**Question 8.** How do you attach a help screen to a screen field or to a screen?

**Question 9.** In the HP ALLBASE/4GL screen painter, how do you initiate line drawing mode?

---

## Answers

**Answer 1.** A validation table can contain 51 values. If you require more than 51 values, you can use the CHECK logic command in an after function on the field to validate the entered data against a number of tables.

**Answer 2.** The scroll area of a screen is defined on the screen header screen. You can specify the first and last line of the scroll area, whether scrolling moves up or down, and a list of data items to be scrolled. You can only use a scroll area on a data screen. They are not available to windows and menus.

**Answer 3.** HP ALLBASE/4GL moves data from the default data movement field to the screen field on display if the screen field buffer is blank. In this particular situation, "000" is moved to the option screen field to allow the user to add option number 000 for a new product.

**Answer 4.** The screen header screen contains a field called *Window Starting Line Number*. You specify the starting line of the window in this field. The window is displayed from this line to the bottom of the screen.

**Answer 5.** Function keys can contain any of the following items as an action. Your answer should include three of the following:

- A process.
- A function.
- A help screen.
- An internal HP ALLBASE/4GL action.
- Another function key set.
- A report.
- A screen.
- An external program.
- A background process.
- Nothing.

**Answer 6.** The SCROLL command displays data on the appropriate scroll line of the current screen. It displays data on the first line of the scroll area if the

## 9-66 Learning Additional Screen Techniques

scroll direction is down, and on the last line of the scroll area if the scrolling direction is up.

**Answer 7.** A select list is a “virtual” HP ALLBASE/SQL table. It contains columns from one or more base tables or view, and each column is defined by a dictionary field specification. Column values can be computed, a constant, or taken directly from a base table column. Select lists are dictionary items in HP ALLBASE/4GL.

**Answer 8.** If a screen field is also a field specification, you can link the help screen to the field using the *Help Name* field on the field specifications screen. You will need to regenerate any screen currently referring to the field specification.

You can also link a help screen to a screen field on the screen field details screen if the field is not a field specification, or if you don't wish the help screen to be tied to every occurrence of the field specification on a screen.

To link a help screen to a screen, enter the name of the help screen in the *Screen Help Name* field on the screen header screen.

**Answer 9.** To initiate line drawing mode from the screen painter, display the Main Keys function key set, and press the **Special Text** function key. Then press the **Line Drawing** function key.

FINAL TRIM SIZE : 7.0 in x 8.5 in



## Expanding Logic Blocks

---

In this chapter you will create a number of logic blocks for the application and alter a few existing logic blocks. These logic blocks include the *product\_proc* process, the functions that read the files, and the data manipulation functions.

You will also create a decision table. Decision tables allow you to define a matrix of conditions and actions. You will use a decision table to determine which file manipulation function should be executed, according to the values of certain variables.

---

### Lesson 24 - Extending the Application Logic

In this lesson you will alter the *product\_proc* process and the screen field logic function *product\_key\_read*. You will also create several new logic blocks, including the *option\_key\_read* function, which is called as an after function on the *option\_no* field on the *option* window.

---

### Objectives

When you have completed this lesson, you will have learned how to:

- Alter processes
- Create new logic blocks

## Modifying the product\_proc Process

Although this process is slightly different for each data manager, its function is the same for all data managers.

### To modify the logic block:

1. Modify the *product\_proc* process as shown below for your data manager and regenerate the process.
2. Then turn to the description of the product on the following pages.
  - a. **KSAM Based Applications**

```
1 MODE *WRITE product option
2 MOVE C-product V-current_record
3 MOVE C-no_record V-product_status
4 TRANSACT *BEGIN
5 SCREEN product_scrn
6 DECISION prod_opt_update
7 IF *IOSTATUS <> "00000" & *IOSTATUS <> N-record_not_found
   THEN ENTER 10
8 TRANSACT *END
9 ENTER 4
10 MESSAGE no_transact
11 TRANSACT *UNDO
12 SCREEN main
```

- b. **HP ALLBASE/SQL Based Applications**

```
1 MODE *WRITE product option
2 MOVE C-product V-current_record
3 MOVE C-no_record V-product_status
4 TRANSACT *BEGIN
5 SCREEN product_scrn
6 DECISION prod_opt_update
7 IF *IOSTATUS <> "00000" & *IOSTATUS <> N-end_of_file
```

## 10-2 Expanding Logic Blocks

```

THEN ENTER 11
8 SQL commit
9 TRANSACT *END
10 ENTER 4
11 MESSAGE no_transact
12 TRANSACT *UNDO
13 SCREEN main

```

### c. HP TurboIMAGE/iX Based Applications

```

1 DM IMAGE *MODE *MODLOCK :D-traindb
2 MODE *UNLOCK product option
3 MOVE C-product V-current_record
4 MOVE C-no_record V-product_status
5 TRANSACT *BEGIN
6 SCREEN product_scrn
7 DECISION prod_opt_update
8 IF *IOSTATUS <> "00000" & *IOSTATUS <> N-record_not_found
   THEN ENTER 12
9 TRANSACT *END
10 DM IMAGE *UNLOCK :D-traindb
11 ENTER 5
12 MESSAGE no_transact
13 SCREEN main

```

## Understanding the Modifications to the Logic

The new and altered steps in this process are described below.

### ■ The altered MODE command.

This step now sets the mode for both the product and option files.

For the HP TurboIMAGE/iX based application, the \*UNLOCK mode is now used. This is necessary because the FILE \*BUFFER commands in the functions you will create later release all locks held on the *traindb* database in \*WRITE mode. This would cause HP ALLBASE/4GL to attempt file operations with no lock placed on a data set, which the DM IMAGE

\*MODLOCK command does not allow. Using MODE \*UNLOCK, locks are only released when a FILE \*UNLOCK command or a DM IMAGE \*UNLOCK is executed.

■ **The new MOVE command.**

This step ensures that the current record type is a product record when the screen is first displayed. Options can only be accessed *after* an existing product has been retrieved from the product file.

■ **The TRANSACT commands.**

This logic block introduces the TRANSACT command. The TRANSACT command defines groups of file operations that make up one logical transaction.

In this application, adding new products and deleting existing products are transactions that involve changes to two data files. To ensure logical consistency of the data files, both files must be updated successfully during the transaction. If a system failure occurs in the time between the two files being updated, the data files could be left in a mutually inconsistent state. In this type of situation, you can use the TRANSACT command to mark the beginning and end of the logical transaction. The TRANSACT \*BEGIN command marks the beginning of the transaction, and the TRANSACT \*END command marks the end.

The TRANSACT \*UNDO command reverses all file transactions that have occurred since the last TRANSACT \*BEGIN command. For transactions that involve HP TurboIMAGE/iX data sets, the TRANSACT

- UNDO command does *NOT* reverse the transactions, so the command is not included in the version of this process used by HP TurboIMAGE/iX application developers.

■ **The DECISION command.**

The DECISION command executes a decision table. The decision table tests the update mode and record type, and then invokes the appropriate file update function. This replaces the previous version of the *product\_proc* process, where separate logic lines were used to make a decision. For situations where there are quite a number of outcomes, a decision table is faster and tidier than using a large number of logic lines.

#### **10-4 Expanding Logic Blocks**

■ **The altered IF command.**

This command is only changed for the KSAM based application. This command still checks to see whether an error occurred. If no error occurs or the record is not found, HP ALLBASE/4GL moves to the next step. The new step at step 9 passes control back to step 4 of the process. If an error, other than a *record not found* error, occurs, control passes to step 10 to the error handling steps.

■ **The altered MESSAGE command.**

Control only passes to this step if an error is detected at step 7 (step 8 for HP TurboIMAGE/iX based applications). This step displays a message to warn the user that an error has occurred and the current product information may be corrupted. The *file\_error* message has been replaced by a more specific warning message.

**To create the accompanying message:**

1. For KSAM based and HP ALLBASE/SQL based applications, create the following message.
2. For HP TurboIMAGE/iX based applications, create the same message, *without* the last line of the message. The last line is not necessary because HP TurboIMAGE/iX file transactions cannot be reversed within HP ALLBASE/4GL.

a. **Message - no\_transact**

|          |                                                                       |
|----------|-----------------------------------------------------------------------|
| Name     | no_transact                                                           |
| Type     | WARN                                                                  |
| Contents | "A file error has occurred. "<br>"This transaction will be reversed." |

---

## Modifying the `product_key_read` Function

This function was previously called as an after function on the `product_no` field on the `product_scrn` screen. When you updated the screen field details, you specified that this function is now called both prior to and after data entry in the `product_no` field.

Seven steps have been added to the start of the function. These steps are executed before the user enters data into the field. They test the current record type, and perform some initialization tasks.

The other part of the function is the “after” function. This part of the function reads the `product` file to retrieve a record matching the product number entered by the user. This part of the function has only altered slightly.

For KSAM based applications, the altered function, and a description of the changes, is listed below. Alter the function and then generate it.

For HP ALLBASE/SQL based applications, turn to HP ALLBASE/SQL Based Applications for details of the function and a description of it.

For HP TurboIMAGE/iX based applications, turn to HP TurboIMAGE/iX Based Applications for details of the function and a description of it.

### To alter the KSAM Based Application Logic:

1. Insert the first seven steps at the start of the `product_key_read` function.
2. Then insert step 11, and alter step 17.
3. Finally, generate the function.

The details of the new steps in the function are explained after the listing.

**Function** - `product_key_read`

```
1 IF *ENTERED *ON THEN ENTER 8
2 IF V-current_record = C-product
   THEN WINDOW prod_wnd; EXIT
3 SHOW *REFRESH S-product_no S-lead_time
4 MATH F-option_no.option + 1 = F-option_no.option
5 SHOW *REFRESH S-option_no
```

### 10-6 Expanding Logic Blocks

```

6 MOVE "5" *FIELDNO
7 EXIT
8 FILE *READ product *KEY= * ; ENTER 13
9 SHOW *REFRESH S-product_no S-lead_time
10 MOVE C-record V-product_status
11 ON 1
12 EXIT
13 IF *IOSTATUS <> N-record_not_found
    THEN MESSAGE file_error ; SERIES 14 16 ; EXIT
14 FILE *BUFFER product
15 CLEAR *MAP S-description S-lead_time
16 MOVE C-no_record V-product_status
17 IF V-mode <> C-add THEN MESSAGE no_product ELSE
    MESSAGE add_prod; VISIT new_option; TIE 2
18 EXIT

```

The operation of this function is described below.

Lines 1 to 7 of the function form the *before* part of the function. The following notes describe the operation of these steps.

- Line 1            This step tests the current state of the switch \*ENTERED. This indicates whether the function is being executed before or after data entry to the field. If the function is being performed after data entry, control passes to step 8. Steps 2 to 7 are only executed when the function is called before data entry in the field.
- Line 2            This step tests the current record type. If it is a *product* record, the current window area is cleared by overlaying the *prod\_wnd* window on the current screen. Then the function exits allowing normal input processing of the field to continue.
- This is always done the very first time the screen is displayed. The *prod\_wnd* window completes the line drawing character framing around the product fields and displays the heading lines for the scroll area. To the user, the screen and the window appear to be a single screen.

- Line 3            This step, and the next four, are only executed if the current record type is an *option*. In this step the product screen fields are redisplayed from the current product record buffer. Since the user can only access an option when a current record exists, the current product record must contain valid data.
- If the current record type is an option record, the option window must already be displayed on the current screen, since the user has accessed an option.
- Line 4            This step increments the current option number by 1.
- Line 5            This step refreshes the option number field on the option window.
- Line 6            This step moves the number of the *Option Number* field to the communications area field \*FIELDNO. When this function exits, the screen processor immediately passes control to that field. Refer to the screen processing rules in Chapter 7 if you want to review this.
- The *Option Number* field is field number 1 of the window on which it was painted. However, when HP ALLBASE/4GL displays a window, the window field numbering is adjusted to start after the number of the last field on the original data screen. In this case field number 1 on the window becomes field number 5 on the base screen.
- Line 7            The function exits at this point, completing the *before* function processing. Processing of the current field finishes, and the cursor moves to the *Option Number* field on the window.
- Control passes to the second part of the function when the function is executed after data entry in the field. Refer to page 9-39 to look back at this part of the function.
- Line 11           User switch 1 is set *on*. This switch is used by the scrolling system to determine if any scrolling has already been done for this product.
- Line 17           This step contains an extra step: **VISIT** new\_option. This function displays the *new\_option* window to allow the user to enter the details of option 000 for a new product.

## 10-8 Expanding Logic Blocks



Now turn to Function - *new\_option*, to continue developing the application.

**To modify the HP ALLBASE/SQL Logic:**

1. Insert the first seven steps at the start of the *product\_key\_read* function.
2. Insert step 12.
3. Alter step 18.
4. Generate the function.

The details of the new steps in the function are explained after the listing.

```
1 IF *ENTERED *ON THEN ENTER 8
2 IF V-current_record = C-product
   THEN WINDOW prod_wnd ; EXIT
3 SHOW *REFRESH S-product_no S-lead_time
4 MATH F-option_no.option + 1 = F-option_no.option
5 SHOW *REFRESH S-option_no
6 MOVE "5" *FIELDNO
7 EXIT
8 SQL find_prod
9 FILE *NEXT product ; ENTER 14
10 SHOW *REFRESH S-product_no S-lead_time
11 MOVE C-record V-product_status
12 ON 1
13 EXIT
14 IF *IOSTATUS <> N-end_of_file
   THEN MESSAGE file_error ; SERIES 15 17 ; EXIT
15 FILE *BUFFER product
16 CLEAR *MAP S-description S-lead_time
17 MOVE C-no_record V-product_status
18 IF V-mode <> C-add THEN MESSAGE no_product
   ELSE MESSAGE add_prod; VISIT new_option; TIE 2
19 EXIT
```

Lines 1 to 7 of the function form the *before* part of the function. The following notes describe the operation of these steps.

- Line 1            This step tests the current state of the switch \*ENTERED. This indicates whether the function is being executed before or after data entry to the field. If the function is being performed after data entry, control passes to step 8. Steps 2 to 7 are only executed when the function is called before data entry in the field.
- Line 2            This step tests the current record type. If it is a *product* record, the current window area is cleared by overlaying the *prod\_wnd* window on the current screen. Then the function exits allowing normal input processing of the field to continue.
- This is always done the very first time the screen is displayed. The *prod\_wnd* window completes the line drawing character framing around the product fields and displays the heading lines for the scroll area. To the user, the screen and the window appear to be a single screen.
- Line 3            This step, and the next four, are only executed if the current record type is an *option*. In this step the product screen fields are redisplayed from the current product record buffer. Since the user can only access an option when a current record exists, the current product record must contain valid data.
- If the current record type is an option record, the option window must already be displayed on the current screen, since the user has accessed an option.
- Line 4            This step increments the current option number by 1.
- Line 5            This step refreshes the option number field on the option window.
- Line 6            This step moves the number of the *Option Number* field to the communications area field \*FIELDNO. When this function exits, the screen processor immediately passes control to that field. Refer to the screen processing rules in Chapter 7 if you want to review this.
- The *Option Number* field is field number 1 of the window on which it was painted. However, when HP ALLBASE/4GL displays a window, the window field numbering is adjusted to

## 10-10 Expanding Logic Blocks

start after the number of the last field on the original data screen. In this case field number 1 on the window becomes field number 5 on the base screen.

Line 8           The function exits at this point, completing the *before* function processing. Processing of the current field finishes, and the cursor moves to the *Option Number* field on the window.

Control passes to the second part of the function when the function is executed after data entry in the field. Refer to chapter 7 to look back at this part of the function.

Line 12           User switch 1 is set *on*. This switch is used by the scrolling system to determine if any scrolling has already been done for this product.

Line 18           This step contains an extra step: **VISIT** *new\_option*. This function displays the *new\_option* window to allow the user to enter the details of option 000 for a new product.

Now turn to Function - *new\_option*, to continue developing the application.

**To modify HP TurboIMAGE/iX logic:**

1. Insert the first seven steps at the start of the *product\_key\_read* function.
2. Insert step 12.
3. Alter step 18.
4. Generate the function.

The details of the new steps in the function are explained after the listing.

**Function** - *product\_key\_read*

```
1 IF *ENTERED *ON THEN ENTER 8
2 IF V-current_record = C-product
   THEN WINDOW prod_wnd ; EXIT
3 SHOW *REFRESH S-product_no S-lead_time
4 MATH F-option_no.option + 1 = F-option_no.option
5 SHOW *REFRESH S-option_no
6 MOVE "5" *FIELDNO
```

```

7 EXIT
8 DM IMAGE *LOCK :D-traindb :R-product
9 FILE *READ product *KEY= * ; ENTER 14
10 SHOW *REFRESH S-product_no S-lead_time
11 MOVE C-record V-product_status
12 ON 1
13 EXIT
14 IF *IOSTATUS <> N-record_not_found
    THEN MESSAGE file_error ; SERIES 15 17 ; EXIT
15 FILE *BUFFER product
16 CLEAR *MAP S-description S-lead_time
17 MOVE C-no_record V-product_status
18 IF V-mode <> C-add THEN MESSAGE no_product
    ELSE MESSAGE add_prod; VISIT new_option; TIE 2
19 EXIT

```

Lines 1 to 7 of the function form the *before* part of the function. The following notes describe the operation of these steps.

- Line 1            This step tests the current state of the switch *\*ENTERED*. This indicates whether the function is being executed before or after data entry to the field. If the function is being performed after data entry, control passes to step 8. Steps 2 to 7 are only executed when the function is called before data entry in the field.
- Line 2            This step tests the current record type. If it is a *product* record, the current window area is cleared by overlaying the *prod\_wnd* window on the current screen. Then the function exits allowing normal input processing of the field to continue.
- This is always done the very first time the screen is displayed. The *prod\_wnd* window completes the line drawing character framing around the product fields and displays the heading lines for the scroll area. To the user, the screen and the window appear to be a single screen.
- Line 3            This step, and the next four, are only executed if the current record type is an *option*. In this step the product screen fields are redisplayed from the current product record buffer. Since

## 10-12 Expanding Logic Blocks

the user can only access an option when a current record exists, the current product record must contain valid data.

If the current record type is an option record, the option window must already be displayed on the current screen, since the user has accessed an option.

- Line 4 This step increments the current option number by 1.
- Line 5 This step refreshes the option number field on the option window.
- Line 6 This step moves the number of the *Option Number* field to the communications area field \*FIELDNO. When this function exits, the screen processor immediately passes control to that field. Refer to the screen processing rules in Chapter 7 if you want to review this.

The *Option Number* field is field number 1 of the window on which it was painted. However, when HP ALLBASE/4GL displays a window, the window field numbering is adjusted to start after the number of the last field on the original data screen. In this case field number 1 on the window becomes field number 5 on the base screen.

- Line 7 The function exits at this point, completing the *before* function processing. Processing of the current field finishes, and the cursor moves to the *Option Number* field on the window.
- Control passes to the second part of the function when the function is executed after data entry in the field. Refer to page 9-39 to look back at this part of the function.

- Line 8 User switch 1 is set *on*. This switch is used by the scrolling system to determine if any scrolling has already been done for this product.

- Line 9 This step contains an extra step: **VISIT** *new\_option*. This function displays the *new\_option* window to allow the user to enter the details of option 000 for a new product.

This function is executed by a VISIT command in the *product\_key\_read* function. It is called when the user is adding

a new product. It displays the window *new\_option* to allow the user to enter the details of option 000 for a new product. Since the *new\_option* window contains required fields, the user cannot commit the current screen without completing the option details.

**To create the new\_option function:**

1. Create this function using the following commands.
2. Generate the function.

```
1 MOVE C-no_record V-option_status
2 FILE *BUFFER option
3 WINDOW new_option
4 EXIT
```

The *new\_option* function is described below.

- |        |                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------|
| Line 1 | This step sets the variable <i>V-option_status</i> to indicate that there is no current record. |
| Line 2 | The FILE *BUFFER command clears the <i>option</i> file record buffer.                           |
| Line 3 | This step displays the <i>new_option</i> window on the <i>product</i> screen.                   |
| Line 4 | The function exits.                                                                             |

**Creating the option\_key\_read Function**

The *option\_key\_read* function is called as an after function on the *option\_no* field on the option window. The function performs some preliminary checking and initialization, and then reads the *option* file or table for a record matching the option number entered by the user.

If you are creating the KSAM based application, continue reading below.

If you are creating the HP ALLBASE/SQL based application, turn to “To create the HP ALLBASE/SQL function”.

**10-14 Expanding Logic Blocks**

If you are creating the HP TurboIMAGE/iX based application, turn to “To create the HP TurboIMAGE/iX function”.

**To create the KSAM function:**

1. Create this function now, using the description below.
2. When finished, generate the function.

```
1 IF V-current_record = C-product THEN EXIT
2 IF V-mode = C-delete & * = N-zero
   THEN MESSAGE del_prod_opt; EXIT
3 LINK 2 *S01 * F-option_key.option
4 FILE *READ option ; ENTER 8
5 SHOW *REFRESH S-option_no S-unit_measure
6 MOVE C-record V-option_status
7 EXIT
8 IF *IOSTATUS <> N-record_not_found
   THEN MESSAGE file_error ; SERIES 9 11 ; EXIT
9 FILE *BUFFER option
10 CLEAR *S 6 9
11 MOVE C-no_record V-option_status
12 IF V-mode <> C-add THEN MESSAGE opt_not_exist
   ELSE MESSAGE add_option
13 EXIT
```

The function operates as described below.

Line 1            This ensures that the function is not performed if the current record being handled is a *product* record. There is only one situation where this may occur. This situation was described when you defined the functions associated with the function keys.

Line 2            This step tests the current mode, and the contents of the *option\_no* field. If the mode is *delete* and the option number is 000, the user is attempting to delete option number zero. This is not permitted, so this step displays an error message.

- Line 3            This step introduces a number of new items. The LINK command concatenates a number of fields into another field. In this example, the LINK command concatenates the fields *\*S01* and *\** into the *option\_key* field on the *option* file buffer. This step builds a unique primary key to access the option file.
- In this command, *\** is a reference to the contents of the current screen field.
- Similarly, *\*S01* is a reference to the contents of the first field on the current screen. On this particular screen it is the *product\_no* field.
- The window for the LINK command does not prompt you for the number of items to be linked together. HP ALLBASE/4GL calculates the number of link fields you enter, and then automatically inserts the correct link count value. The LINK command also has an optional joiner character which is inserted between each field as it is linked. This example does not need a joiner character so you can leave the field blank.
- Line 4            This step reads the option file. Since the previous step moved the key value to the appropriate field in the file buffer, there is no need to specify the key for this file read operation. In all other respects this command is identical to the file read statements that you have already used.
- Line 5            If the specified record is found, the SHOW command displays the record buffer contents in the appropriate fields on the screen.
- Line 6            An option record has been found, so the *option\_status* variable is set to reflect this.
- Line 7            The function exits.
- Line 8            This step starts the file read error section of the function. If the file error is caused by any condition other than a record not found condition, it displays an error message, clears the screen and file buffers, sets the status variable, and then exits.
- Line 9            This step clears the option file buffer.

## 10-16 Expanding Logic Blocks



- Line 10            This step clears the option window fields.
- Line 11            The *option\_status* variable is set to show that a record was not successfully read.
- Line 12            The mode is checked and an appropriate message is displayed.
- Line 13            The function exits.

Now turn to “Creating Messages for the Logic Blocks” to continue developing your application.

**To create the HP ALLBASE/SQL function:**

1. Create this function using the description below.
2. When finished, generate the function.

```

1 IF V-current_record = C-product THEN EXIT
2 IF V-mode = C-delete & * = N-zero THEN
  MESSAGE del_prod_opt ; EXIT
3 MOVE * V-option_no
4 SQL option_key_sel
5 FILE *NEXT option ; ENTER 9
6 SHOW *REFRESH S-option_no S-unit_measure
7 MOVE C-record V-option_status
8 EXIT
9 IF *IOSTATUS <> N-end_of_file THEN MESSAGE file_error ;
  SERIES 10 12 ; EXIT
10 FILE *BUFFER option
11 CLEAR *S 6 9
12 MOVE C-no_record V-option_status
13 IF V-mode <> C-add THEN MESSAGE opt_not_exist
  ELSE MESSAGE add_option
14 EXIT

```

- Line 1            This ensures that the function is not performed if the current record being handled is a *product* record.

There is only one situation where this may occur. When a user presses the **Product** function key, this action causes a field commit for the current field. In this case, the function called from the function key sets V-current\_record to C-product so that the *option\_key\_read* function is not fully completed.

- Line 2 This step tests the current mode, and the contents of the *option\_no* field. If the mode is *delete* and the option number is 000, the user is attempting to delete option number zero. This is not permitted, so this step displays an error message.
- Line 3 This step moves the value in the current field (the *option\_no* field on the option window) to the variable *V-option\_no*. The SQL logic block *option\_key\_sel* that is called by step 4 uses this variable.
- Line 4 This step calls the SQL logic block *option\_key\_sel*. You will define this SQL logic block after completing the *option\_key\_read* function.
- Line 5 This step reads the next record in the option file.
- Line 6 If the specified record is found, the SHOW command displays the record buffer contents in the appropriate fields on the screen.
- Line 7 An option record has been found, so the *option\_status* variable is set to reflect this.
- Line 8 The function exits.
- Line 9 This step starts the file read error section of the function. If the file error is caused by any condition other than an end of file condition, it displays an error message, clears the screen and file buffers, sets the status variable, and then exits.
- Line 10 This step clears the option file buffer.
- Line 11 This step clears the option window fields.
- Line 12 The *option\_status* variable is set to show that a record was not successfully read.
- Line 13 The mode is checked and an appropriate message is displayed.

## 10-18 Expanding Logic Blocks

Line 14            The function exits.

This application does not require a unique key for the option table since the *opt\_key\_sel* SQL logic block uses a combined search condition to find the required record in the *option* table.

**To create the option\_key\_sel logic block:**

1. Create this SQL logic block as shown below.
2. Then generate the logic block.

```
SELECT :option FROM sqlgrp.option
  WHERE product_no = :F-product_no.product
  AND option_no = :V-option_no
FOR UPDATE OF product_no, option_no, description, cost,
  quantity, unit_measure;
```

This SQL logic block declares and opens a cursor on the *option* table. The SQL logic block also shows the use of an AND clause to specify a compound search condition. In this case, the SELECT command locates a row in the table with the *product\_no* field equal to the current value in the *product\_no* field on the *product* file buffer, and the *option\_no* field equal to the value in the variable *V-option\_no*. The FILE \*NEXT command at step 5 of the *option\_key\_read* function retrieves the first record in the active set for the cursor.

This example demonstrates the host variable referencing technique for SQL logic blocks. In this SQL logic block, the term *:F-product\_no.product* is a reference to the field *product\_no* on the file record buffer *product*, and the term *:V-option\_no* is a reference to the variable *option\_no*. Note that you must use the full reference for the data item, and prefix the reference with a colon (:).

Now turn to “Creating Messages for the Logic Block,” to continue developing your application.

**To create the HP TurboIMAGE/iX function:**

1. Create the function using the description shown below.
2. Then generate the function.

```

1 IF V-current_record = C-product THEN EXIT
2 IF V-mode = C-delete & * = N-zero
   THEN MESSAGE del_prod_opt; EXIT
3 DM IMAGE *LOCK :D-traindb :R-option
4 LINK 2 *S01 * F-option_key.option
5 FILE *READ option ; ENTER 9
6 SHOW *REFRESH S-option_no S-unit_measure
7 MOVE C-record V-option_status
8 EXIT
9 IF *IOSTATUS <> N-record_not_found
   THEN MESSAGE file_error ; SERIES 10 12 ; EXIT
10 FILE *BUFFER option
11 CLEAR *S 6 9
12 MOVE C-no_record V-option_status
13 IF V-mode <> C-add THEN MESSAGE opt_not_exist
   ELSE MESSAGE add_option
14 EXIT

```

The function operates as described below.

- |        |                                                                                                                                                                                                                                                                  |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This ensures that the function is not performed if the current record being handled is a <i>product</i> record. There is only one situation where this may occur. This situation was described when you defined the functions associated with the function keys. |
| Line 2 | This step places an HP TurboIMAGE/iX logical lock on the <i>option</i> data set. This lock is released in the <i>product_proc</i> process.                                                                                                                       |
| Line 3 | This step tests the current mode, and the contents of the <i>option_no</i> field. If the mode is <i>delete</i> and the option number is 000, the user is attempting to delete option number zero. This is not permitted, so this step displays an error message. |
| Line 4 | This step introduces a number of new items. The LINK command concatenates a number of fields into another field. In this example, the LINK command concatenates the fields <i>*S01</i>                                                                           |

## 10-20 Expanding Logic Blocks

and \* into the *option\_key* field on the *option* file buffer. This step builds a unique primary key to access the option file.

In this command, \* is a reference to the contents of the current screen field.

Similarly, \*S01 is a reference to the contents of the first field on the current screen. On this particular screen it is the *product\_no* field.

The window for the LINK command does not prompt you for the number of items to be linked together. HP ALLBASE/4GL calculates the number of link fields you enter, and then automatically inserts the correct link count value. The LINK command also has an optional joiner character which is inserted between each field as it is linked. This example does not need a joiner character so you can leave the field blank.

- Line 5 This step reads the option data set. Since the previous step moved the key value to the appropriate field in the file buffer, there is no need to specify the key for this file read operation. In all other respects this command is identical to the file read statements that you have already used.
- Line 6 If the specified record is found, the SHOW command displays the record buffer contents in the appropriate fields on the screen.
- Line 7 An option record has been found, so the *option\_status* variable is set to reflect this.
- Line 8 The function exits.
- Line 9 This step starts the file read error section of the function. If the file error is caused by any condition other than a record not found condition, it displays an error message, clears the screen and file buffers, sets the status variable, and then exits.
- Line 10 This step clears the option file buffer.
- Line 11 This step clears the option window fields.
- Line 12 The *option\_status* variable is set to show that a record was not successfully read.

Line 13            The mode variable is checked and an appropriate message is displayed.

Line 14            The function exits.

You have now completed the screen field functions.

---

## Creating Messages for the Logic Blocks

This page describes the messages required by the logic blocks you have just created.

1. Use the *Messages* selection on the dictionary menu to create these messages now.

- a. **Add Option.** The *option\_key\_read* function displays this message to tell the user that the details for a new option may be added.

|          |                                           |
|----------|-------------------------------------------|
| Name     | add_option                                |
| Type     | MESS                                      |
| Contents | "New option, number " * ", may be added." |

- b. **Option Doesn't Exist.** The *option\_key\_read* function displays this message to indicate that a record for the option number does not exist.

|          |                                                     |
|----------|-----------------------------------------------------|
| Name     | opt_not_exist                                       |
| Type     | ERROR                                               |
| Contents | "Record does not exist for option number"<br>* ". " |

- c. **Delete Product Options.** The *option\_key\_read* function displays this message. A product cannot be deleted unless the options for the product are deleted first.

|          |                                                            |
|----------|------------------------------------------------------------|
| Name     | del_prod_opt                                               |
| Type     | ERROR                                                      |
| Contents | "To delete option 000, "<br>"you must delete the product." |

### 10-22 Expanding Logic Blocks

Now all of the screen and window handling for this application is complete. You now need to provide the ability to update the application data files. The first item to define is a decision table to invoke the correct update function.

If you wish, you can use the *Test* option on the developer main menu to look at the new *product\_scrn* screen. However, none of the file manipulation functions or the scrolling functions will be operational yet.

---

## Summary

In this lesson you created a number of logic blocks for the application.

Some of these logic blocks introduced logic commands that you haven't used before. The commands are:

- The TRANSACT command.

This command allows you to define groups of file transactions that must be performed together to ensure logical consistency of the files. The command exists in three forms. The \*BEGIN and \*END forms of the command mark the start and finish of the group of file commands. The \*UNDO form of the command reverses all KSAM and HP ALLBASE/SQL file transactions that have occurred since the last TRANSACT \*BEGIN command.

- The LINK command (not used in this HP ALLBASE/SQL application).

The LINK command concatenates a number of fields, and places the result in a specified field. The LINK command optionally allows you to specify a “joiner” that is inserted between the fields as they are linked.

In this application, you used the LINK command to concatenate the product number and the option number to create a unique *option\_key* to read the option file.

The *HP ALLBASE/4GL Developer Reference Manual* describes these logic commands in more detail.

---

## Lesson 25 - Using Decision Tables

Decision tables allow you to define a series of actions to be performed as the result of the outcome of a series of conditional tests or questions. A decision table can test up to eight questions and execute up to eight actions such as functions, processes and screens.

Having defined the questions and actions, you can then define up to 31 relationships. Each relationship defines a combination of results for the questions, the actions to be performed, and the order in which the actions are performed.

You can use a decision table to perform the same functions as complex IF-THEN-ELSE structures.

A decision table is defined with four screens:

- The decision table header screen.
- The decision table questions screen.
- The decision table actions screen.
- The decision table relationships screen.

In this application, the decision table tests the status of a number of variables that are set while the *product* screen is active. The results of these tests determine the appropriate file update actions.

The decision table you will create is used by the *product\_proc* process. The process executes this decision table when the user terminates the *product\_scrn* screen. The decision table determines which update function is executed to update the product and option files after data entry.

You will also create the functions that add new records to the product and option files.

### 10-24 Expanding Logic Blocks



---

## Objectives

When you have completed this lesson you will have learned how to:

- Create a decision table.
- Create functions to add new records to the product and option files.

---

## Defining the Decision Table Header

You must define a decision table header before you can access any other parts of the decision table.

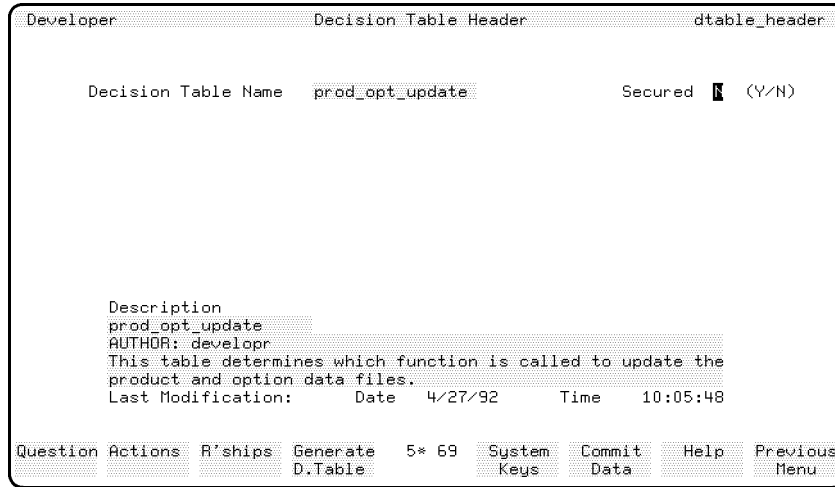
### Menu Path

#### To access the decision table header screen:

1. From the main menu, select the *Logic* option.
2. Select the *Decision Tables* option.
3. Choose *Header*.

### Screen Description

This screen allows you to enter brief details about the decision table.



### Decision Table Header Screen

To enter the field values:

1. Make the screen entries indicated in the table below.

| Field               | Entry                         | Explanation                                                                                             |
|---------------------|-------------------------------|---------------------------------------------------------------------------------------------------------|
| Decision Table Name | prod_opt_update               | This is the name of the decision table.                                                                 |
| Secured             | Accept the default.           | This indicates whether the decision table is secured against modification by an unauthorized developer. |
| Description         | Enter a suitable description. |                                                                                                         |

2. Press the **Commit Data** function key to create the header for the decision table.

### 10-26 Expanding Logic Blocks

---

## Decision Table Questions

### Menu Path

#### To access the Decision Table Questions Screen

1. From the main menu, select the *Logic* option.
2. Select the *Decision Tables* option.
3. Choose *Questions*.

#### Screen Description

This screen allows you to enter up to eight separate IF statements. These are the questions to be resolved when the decision table is executed.

The screenshot shows a terminal window titled "Decision Table Questions" with the following content:

```
Developer          Decision Table Questions          dtablequestions

      Decision Table      prod_opt_update          Secured  N (Y/N)

Questions
IF Data name 1:          U-mode          IF Test   =
Data name 2:            C-add
IF Data name 1:          U-mode          IF Test   =
Data name 2:            C-modify
IF Data name 1:          U-mode          IF Test   =
Data name 2:            C-delete
IF Data name 1:          U-current_record  IF Test   =
Data name 2:            C-product
IF Data name 1:          U-current_record  IF Test   =
Data name 2:            C-option
IF Data name 1:
Data name 2:
IF Data name 1:
Data name 2:
IF Data name 1:
Data name 2:
IF Data name 1:
Data name 2:
```

At the bottom of the screen, there is a navigation bar with the following options: Header, Actions, R'ships, Generate D.Table, 4\* 25, System Keys, Commit Data, Help, and Previous Menu.

### Decision Table Questions Screen

**To enter the field values:**

| <b>Field</b>               | <b>Entry</b>          | <b>Explanation</b>                                                                                                                                                                                                  |
|----------------------------|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Decision Table Name</b> | Press <b>(Return)</b> | This field defaults to the name used in the last decision table screen. In this case, it defaults to <i>prod_opt_update</i> from the decision table header screen.                                                  |
| <b>Questions</b>           |                       | The remainder of this screen contains eight sets of three fields. The fields are <i>IF Data name 1</i> , <i>IF Test</i> , and <i>Data name 2</i> . Each set of three fields specifies a question to be resolved.    |
| <b>Data name 1</b>         | <b>V-mode</b>         | This is the name of the first data item that is to be tested. The rules for the construction of an IF statement in a decision table are the same as those for the construction of an IF statement in a logic block. |
| <b>IF Test</b>             | <b>=</b>              | This is the relationship that must exist between the two named fields. (A decision table question can also test the status of a single item. For example, the test can be *BLANK.)                                  |
| <b>Data name 2</b>         | <b>C-add</b>          | This is the name of the data item that the first data item is tested against. In this case it is the alphanumeric constant <i>add</i> .                                                                             |

The first question is now complete.

**To complete the screen:**

Enter the remaining four questions as shown below:

**1. Question 2**

|             |          |
|-------------|----------|
| Data Name 1 | V-mode   |
| IF Test     | =        |
| Data Name 2 | C-modify |

**10-28 Expanding Logic Blocks**

2. **Question 3**

|             |          |
|-------------|----------|
| Data Name 1 | V-mode   |
| IF Test     | =        |
| Data Name 2 | C-delete |

3. **Question 4**

|             |                  |
|-------------|------------------|
| Data Name 1 | V-current_record |
| IF Test     | =                |
| Data Name 2 | C-product        |

4. **Question 5**

|             |                  |
|-------------|------------------|
| Data Name 1 | V-current_record |
| IF Test     | =                |
| Data Name 2 | C-option         |

5. Press the **Commit Data** function key when you have completed these entries.

---

## Defining Decision Table Actions

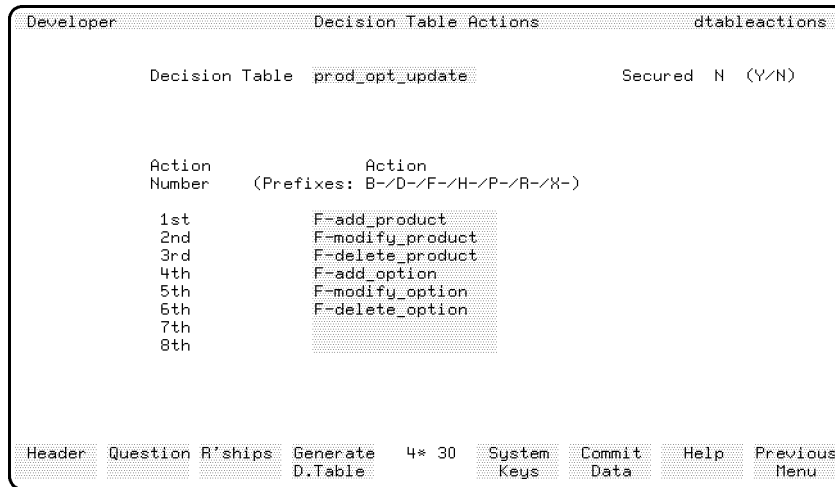
### Menu Path

#### To access the Decision Table Actions Screen:

1. From the main menu, select the *Logic* option.
2. Select the *Decision Tables* options.
3. Choose *Actions*.

#### Screen Description

This screen allows you to specify up to eight actions to be performed as a result of the resolution of the decision table questions.



### Decision Table Actions Screen

To enter the field values:

1. Make the entries listed in the table below.

| Field               | Entry               | Explanation                                                                                                                                                           |
|---------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Decision Table Name | Press <b>Return</b> | This field defaults to the name used in the last decision table screen. In this case, it defaults to <i>prod_opt_update</i> from the decision table questions screen. |
| Actions             |                     | The next eight fields contain the type and name of the action to be performed for the particular action number.                                                       |
| 1st                 | F-add_product       | This indicates that action number 1 is the <i>add_product</i> function. The <i>F-</i> prefix indicates that the action is a function.                                 |
| 2nd                 | F-modify_product    |                                                                                                                                                                       |
| 3rd                 | F-delete_product    |                                                                                                                                                                       |
| 4th                 | F-add_option        |                                                                                                                                                                       |
| 5th                 | F-modify_option     |                                                                                                                                                                       |

### 10-30 Expanding Logic Blocks

2. Press the **Commit Data** function key when you have finished these entries.

---

## Defining Decision Table Relationships

### Menu Path

#### To access the Decision Table Relationships screen:

1. From the main menu, select the *Logic* option.
2. Select the *Decision Tables* option.
3. Choose *Relationships*.

## Screen Description

This screen allows you to connect the various decision table components together. It allows you to define the combinations of questions that result in certain actions being executed.

It contains a series of *Yes*, *No*, or *Don't Care* responses to the questions and an ordered list of the actions to be performed if those results occur.

```

Developer          Decision Table Relationships          dtablerlships

Decision Table      prod_opt_update          Secured  N (Y/N)
Column Number      1          Amendment Mode  A (A/D)

----- Question -----
Number      Response      Possible      Order of
            (Y/N/-)      Action      Execution
            (1-8 or 0 to ignore)

1  ....      Y          F-add_product      1
2  ....      -          F-modify_product   0
3  ....      -          F-delete_product   0
4  ....      Y          F-add_option        0
5  ....      -          F-modify_option     0
6  ....      -          F-delete_option     0
7  ....      -
8  ....      -

Column Definition      Add prod/base option

Header  Question  Actions  Generate  6* 66  System  Commit  Help  Previous
        Question  Actions  D.Table  Keys   Data   Menu
  
```

### Decision Table Relationships Screen

To enter the field values:

1. Make the entries on the screen as shown in the table below:

| Field               | Entry                 | Explanation                                                                                      |
|---------------------|-----------------------|--------------------------------------------------------------------------------------------------|
| Decision Table Name | Press <b>(Return)</b> | This field defaults to <i>prod_opt_update</i> , the name used in the last decision table screen. |

## 10-32 Expanding Logic Blocks



| Field                 | Entry                                                                                                                                                                  | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Column Number</b>  | Accept the default.                                                                                                                                                    | This is the number of the column or set of relationships that you are about to define. You can define up to 31 columns.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Amendment Mode</b> | Accept the default.                                                                                                                                                    | The entry for this field allows you to choose what you want to do with this particular column, <i>A</i> = add or amend, and <i>D</i> = delete.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Questions</b>      | Make the entries listed below.<br><br><b>Number - Response</b><br>1. .... Y<br>2. .... -<br>3. .... -<br>4. .... Y<br>5. .... -<br>6. .... -<br>7. .... -<br>8. .... - | The first column of fields ( <i>Number</i> column) refers to the questions defined on the decision tables questions screen. They are numbered in the same order as they were defined. The second column of fields ( <i>Response</i> column) refers to the answers to the questions. The responses default to -. This means that the outcome of the question has no bearing on the column.<br><br>If you enter <b>Y</b> the question must be resolved as <i>true</i> for the column to be valid. If you enter <b>N</b> the question must be resolved as <i>false</i> for the column to be valid. All eight criteria must be met for the column to be valid.<br><br>This column is resolved as true when both <i>V-mode</i> is equal to <i>C-add</i> and <i>V-current_record</i> is equal to <i>C-product</i> . This is effectively stating that a product record must be added. |

| Field                    | Entry                          | Explanation                                                                                                                                                                                                                                                                        |
|--------------------------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Actions</b>           | Complete the column as listed. | This column specifies the actions that are to be performed, and the order in which they are performed. A set of display fields show the names of the actions you have already defined.                                                                                             |
|                          | <b>Action - Order of Exec.</b> |                                                                                                                                                                                                                                                                                    |
|                          | F-add_product.... 1            |                                                                                                                                                                                                                                                                                    |
|                          | F-modify_product.... 0         | The column to the right (Order of Execution) initially defaults to all 0s.                                                                                                                                                                                                         |
|                          | F-delete_product.... 0         | The default (0) means don't do this action. The numbers specified next to the action indicate which actions are executed if the column resolves as true, and also indicate the order in which the actions are performed if more than action one is executed.                       |
|                          | F-add_option..... 0            |                                                                                                                                                                                                                                                                                    |
|                          | F-modify_option..... 0         |                                                                                                                                                                                                                                                                                    |
|                          | F-delete_option..... 0         |                                                                                                                                                                                                                                                                                    |
|                          | ..... 0                        |                                                                                                                                                                                                                                                                                    |
|                          | .....0                         | The numbers you specify must be contiguous and a number cannot occur more than once. For example, 10204000 is not valid as the 3 is missing. It should read 10203000. Similarly, the entry 00220104 is not valid as the 2 appears twice. It should be either 00230104 or 00320104. |
|                          |                                | This relationship executes the <i>add_product</i> function if the column is resolved as true.                                                                                                                                                                                      |
| <b>Column Definition</b> | Add prod/base option           | This is a comment field that allows you to enter a description of what the column does. It has no operational effect.                                                                                                                                                              |

- When you have entered all of the data, press the **Commit Data** function key.

**To complete the remaining columns for the decision table:**

There are five more columns to be created for this decision table.

- Complete the columns as shown.

**10-34 Expanding Logic Blocks**

a. **Column 2**

| <b>Question</b> | <b>Response</b> | <b>Action</b>    | <b>Order</b> |
|-----------------|-----------------|------------------|--------------|
| 1               | Y               | F-add_product    | 0            |
| 2               | -               | F-modify_product | 0            |
| 3               | -               | F-delete_product | 0            |
| 4               | -               | F-add_option     | 1            |
| 5               | Y               | F-modify_option  | 0            |
| 6               | -               | F-delete_option  | 0            |
| 7               | -               |                  | 0            |
| 8               | -               |                  | 0            |

**Column Definition:** Add the Option

b. **Column 3**

| <b>Question</b> | <b>Response</b> | <b>Action</b>    | <b>Order</b> |
|-----------------|-----------------|------------------|--------------|
| 1               | -               | F-add_product    | 0            |
| 2               | Y               | F-modify_product | 1            |
| 3               | -               | F-delete_product | 0            |
| 4               | Y               | F-add_option     | 0            |
| 5               | -               | F-modify_option  | 0            |
| 6               | -               | F-delete_option  | 0            |
| 7               | -               |                  | 0            |
| 8               | -               |                  | 0            |

**Column Definition:** Modify the Product

c. Column 4

| Question | Response | Action           | Order |
|----------|----------|------------------|-------|
| 1        | -        | F-add_product    | 0     |
| 2        | Y        | F-modify_product | 0     |
| 3        | -        | F-delete_product | 0     |
| 4        | -        | F-add_option     | 0     |
| 5        | Y        | F-modify_option  | 1     |
| 6        | -        | F-delete_option  | 0     |
| 7        | -        |                  | 0     |
| 8        | -        |                  | 0     |

Column Definition: Modify the Option

d. Column 5

| Question | Response | Action           | Order |
|----------|----------|------------------|-------|
| 1        | -        | F-add_product    | 0     |
| 2        | -        | F-modify_product | 0     |
| 3        | Y        | F-delete_product | 1     |
| 4        | Y        | F-add_option     | 0     |
| 5        | -        | F-modify_option  | 0     |
| 6        | -        | F-delete_option  | 0     |
| 7        | -        |                  | 0     |
| 8        | -        |                  | 0     |

Column Definition: Delete the Product

**10-36 Expanding Logic Blocks**

e. **Column 6**

| Question | Response | Action           | Order |
|----------|----------|------------------|-------|
| 1        | -        | F-add_product    | 0     |
| 2        | -        | F-modify_product | 0     |
| 3        | Y        | F-delete_product | 0     |
| 4        | -        | F-add_option     | 0     |
| 5        | Y        | F-modify_option  | 0     |
| 6        | -        | F-delete_option  | 1     |
| 7        | -        |                  | 0     |
| 8        | -        |                  | 0     |

**Column Definition Delete the Option**

This completes the definition of the decision table.

2. Generate the decision table by pressing the **Generate D.Table** function key on the decision table relationships screen.

You have now created most of the necessary dictionary items, data screens, functions and processes to drive the application.

**Testing the Modified Application**

There are a number of further items that you need to create before the application is complete. The remaining items are the file update functions (the logic blocks that actually add, modify, and delete records in the appropriate files), the scrolling functions, some messages, and a report.

However, you can now test part of the application if you wish. You can select any of the modes on the main menu and display the product screen. Since you haven't created the file updating functions yet, you won't be able to perform any valid operations on the files.

---

## Summary

In this lesson you defined a decision table. A decision table is similar to a set of IF-THEN-ELSE statements. You can use a decision table to test a number of conditions and then execute a specified set of actions in a specified sequence depending on the results of the test.

Decision tables have the following features:

- To define a decision table, you must complete four screens. These are the decision table header screen, the decision table questions screen, the decision table actions screen, and the decision table relationships screen.
- The questions for a decision table are in the same format as logic IF statements.
- A decision table can execute any action that can be called from a menu. These include functions, processes, screens, and further decision tables.
- For any given set of outcomes for the decision table questions, the decision table relationships specify which actions are executed, and the sequence in which they are executed.
- A decision table must be generated before it can be used in the application.

---

## Lesson 26 - Creating the File Manipulation Functions

The next task is to modify the functions that allow you to add, modify, and delete products, and to create the functions that allow you to add, modify, and delete options.

---

### Objectives

When you have completed this lesson, you will have learned how to create the logic needed to complete the logic required for this extension of the *training* application.

You will have practiced defining:

- Add functions
- Modify functions
- Delete functions

---

### Defining the Add Functions

The functions that add new records to the *product* and *option* files are called by the *prod\_opt\_update* decision table, which you have just completed.

**Function** - *add\_product*

The *add\_product* function already exists, but requires some modification.

**To modify the add-product function:**

1. Insert the following step at step 5 of the *add\_product* function:

**VISIT** *add\_option*

This step calls the *add\_option* function, which is described below.

2. Then regenerate the function.

**Function** - *add\_option*

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, read on below.

If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications.

### **KSAM Based and HP TurboIMAGE/iX Based Applications**

The *add\_option* function contains the following steps:

```
1 IF V-option_status = C-record THEN
    MESSAGE option_warn ; EXIT
2 MOVE S-product_no F-product_no.option
3 LINKLOOP 2 F-product_no.option 1 F-option_key.option
4 FILE *INSERT option; ENTER 8
5 MESSAGE opt_add_ok
6 IF F-option_no.option = N-zero THEN MOVE C-option V-current_record;
    WINDOW option; MOVE "1" *NEWTIE
7 EXIT
8 MESSAGE file_error
9 EXIT
```

The *add\_option* function operates as follows:

- |        |                                                                                                                                                                                                                                                                                                                                                                                |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | Like the first step of the <i>add_product</i> function, this step makes sure that the option does not already exist. If the option does exist, this step displays a warning message and the function exits.                                                                                                                                                                    |
| Line 2 | This step moves the current product number into the <i>product_no</i> field on the <i>option</i> file buffer. This is necessary because the buffer was cleared before the new data was entered.                                                                                                                                                                                |
| Line 3 | This step builds the unique option file key. There are several ways that this could have been achieved. This particular method uses the LINKLOOP command to build the key.<br><br>Each ... <i>LOOP</i> command performs a repeated operation on successive fields in a file record, screen, or the scratch pad. You must specify the number of times the command is performed, |

### **10-40 Expanding Logic Blocks**



the field that the operation starts on, and the step factor to indicate the next field or fields to be operated on.

In this step, the LINKLOOP command is performed twice, as shown by the first number after the LINKLOOP command. The first field is *F-product\_no.option*. The step factor of 1 indicates that the next field is the next operand. The result is placed in the field *F-option\_key.option*.

The same result could also be achieved with the command:

```
LINKLOOP 2 *S01 4 F-option_key.option
```

This command takes the first screen field as the first operand and links it to a screen field four fields further on. On this particular screen, this is the *option number* field.

- Line 4      The new option is added to the option file. If a file access error is detected, control passes to step 8.
- Line 5      If the file insert operation in step 4 is successful, the user is informed that the file was updated.
- Line 6      If the option number written to the file was 000, the user has just added a new product. The current record type is set to *option*. This step then displays the *option* window and sets the value in the communication area field \*NEWTIE to 1. This is equivalent to the command TIE 1, and passes control to field number 1 on the *product* screen when it is next displayed.
- Line 7      The function exits.
- Line 8      If a file access error was detected in step 4, the user is informed by a message.
- Line 9      The function exits.

Now turn to The Option\_Number Function to continue developing the application.

### **HP ALLBASE/SQL Based Applications**

```
1 IF V-option_status = C-record  
  THEN MESSAGE option_warn ; EXIT
```

```

2 MOVE S-product_no F-product_no.option
3 FILE *INSERT option ; ENTER 7
4 MESSAGE opt_add_ok
5 IF F-option_no.option = N-zero THEN
    MOVE C-option V-current_record ;
    WINDOW option ; MOVE "1" *NEWTIE
6 EXIT
7 MESSAGE file_error
8 EXIT

```

The *add\_option* function operates as follows:

Line 1            Like the first step of the *add\_product* function, this step makes sure that the record does not already exist. If the record does exist, this step displays a warning message and the function exits.

Line 2            This step moves the current product number into the *product\_no* field on the *option* file buffer. This is necessary because the buffer was cleared before the new data was entered.

Line 3            The new option is inserted into the option file. If a file access error is detected, control passes to step 7.

The FILE \*INSERT command in this function inserts the current buffer contents into the *option* table as a new row. The FILE \*INSERT command is equivalent to the following SQL command:

```

INSERT INTO sqlgrp.option
VALUES (:F-product_no.option,
        :F-option_no.option,
        :F-description.option,
        :F-cost.option,
        :F-quantity.option),
        :F-unit_measure.option);

```

Line 4            If the file insert operation in step 3 is successful, the user is informed that the file was updated.

## 10-42 Expanding Logic Blocks

- Line 5            If the option number written to the file was 000, then the user has just added a new product. The current record type is set to *option*. This step then displays the *option* window and sets the value in the communication area field \*NEWTIE to 1. This is equivalent to the command TIE 1, and passes control to field number 1 on the *product* screen when it is next displayed.
- Line 6            The function exits.
- Line 7            If a file access error was detected in step 3, the user is informed by a message.
- Line 8            The function exits.

### Modifying the option\_number Function

The *option\_number* function is called as a prior function from the *option\_no* field on a *new\_option* window. This window is executed when the user enters the details for option number 000 for a new product. The *option\_no* function contains the following commands.

If you are developing the KSAM based application or the HP ALLBASE/SQL based application, continue reading below.

If you are developing the HP TurboIMAGE/iX based application, turn to HP TurboIMAGE/iX Based Applications for instructions.

#### To modify the option-number function for KSAM and HP ALLBASE/SQL Applications:

1. Create the *option\_number* function, using the logic block listed below.
2. Generate the completed function.

```

1 FILE *BUFFER option
2 CLEAR *S 6 8
3 MOVE N-zero F-option_no.option

```

The *option\_no* function operates like this:

- Line 1            This step clears the *option* file buffer of any data from a previous update.

- Line 2            This step clears the screen fields from the *description* field to the *qty\_on\_hand* field.
- Line 3            \*This step moves the value 000 to the *option\_no* field on the *option* file buffer.
3. Turn to “Creating the Appropriate Messages” to continue developing your application.

**To modify the option-number function for HP TurboIMAGE/iX Applications:**

1. Create and generate this function now.

```
1 FILE *BUFFER option
2 CLEAR *S 6 8
3 MOVE N-zero F-option_no.option
4 DM IMAGE *LOCK :D-traindb :R-option
```

The *option\_no* function operates like this:

- Line 1            This step clears the *option* file buffer of any data from a previous update.
- Line 2            This step clears the screen fields from the *description* field to the *qty\_on\_hand* field.
- Line 3            This step moves the value 000 to the *option\_no* field on the *option* file buffer.
- Line 4            This step places an HP TurboIMAGE/iX logical lock on the *option* data set. This lock is released by the *product\_proc* process.

The *option\_key\_read* function normally places a lock on this data set before adding an option record, but that function is not executed for the first option for each product. This function is always executed before the first option for a product is added.

2. Create the messages that accompany this logic.

**10-44 Expanding Logic Blocks**

## Creating the Appropriate Messages

The *add\_option* function needs the following messages.

1. Create these messages.

a. **Message** - *option\_warn*

|          |                                                        |
|----------|--------------------------------------------------------|
| Name     | option_warn                                            |
| Type     | WARN                                                   |
| Contents | "This option already exists."<br>"It cannot be added." |

b. **Message** - *opt\_add\_ok*

|          |                                                                     |
|----------|---------------------------------------------------------------------|
| Name     | opt_add_ok                                                          |
| Type     | MESS                                                                |
| Contents | "New option, number"<br>F-option_no.option<br>", has been created." |

---

## Creating the modify\_option Function

The *modify\_option* function is very similar to the *modify\_product* function you created in lesson 15, chapter 7, so no explanation is provided. You should be able to interpret the function's behavior. The function details for each data manager are listed below and on the following pages.

**To create the function forKSAM Based Applications:**

1. Create and generate this function.

**Function** - *modify\_option*

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option ; EXIT
2 FILE *WRITE option ; ENTER 5
3 MESSAGE opt_modify_ok
4 EXIT
```

```
5 MESSAGE file_error
6 EXIT
```

2. Now turn to the following page to create the messages required by the function.

**To create the function for HP ALLBASE/SQL Based Applications:**

1. Create and generate this function now.

**Function** - *modify\_option*

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option; EXIT
2 SQL modify_option ; ENTER 5
3 MESSAGE opt_modify_ok
4 EXIT
5 MESSAGE file_error
6 EXIT
```

This function calls the SQL logic block *modify\_option*.

**SQL Logic Block** - *modify\_option*

2. Create and generate the *modify\_option* SQL logic block. It modifies a row in the *product* table.

```
UPDATE sqlgrp.option SET
    description = :F-description.option,
    cost        = :F-cost.option,
    quantity    = :F-quantity.option,
    unit_measure = :F-unit_measure.option
WHERE CURRENT OF option_key_sel;
```

This SQL logic block updates the row of the table defined by the present position of the *option\_key\_sel* cursor. The *option\_key\_read* function declares and opens this cursor when it calls the SQL logic block *option\_key\_sel*.

Note that the cursor name is the name of the SQL logic block containing the SELECT command that declares and opens the cursor.

## 10-46 Expanding Logic Blocks

3. Now turn to the Message heading below the following HP TurboIMAGE/iX instructions.

**To complete the function for HP TurboIMAGE/iX Based Applications:**

1. Create and generate this function now.

**Function** - *modify\_option*

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option ; EXIT
2 FILE *WRITE option *INDEX=option_no ; ENTER 5
3 MESSAGE opt_modify_ok
4 EXIT
5 MESSAGE file_error
6 EXIT
```

2. Next, create the messages required for this function.

**To create the messages for the modify\_option function:**

1. Create the following messages.

a. **Message** - *no\_option*

|          |                               |
|----------|-------------------------------|
| Name     | no_option                     |
| Type     | WARN                          |
| Contents | "No option for modification." |

b. **Message** - *opt\_modify\_ok*

|          |                                                                |
|----------|----------------------------------------------------------------|
| Name     | opt_modify_ok                                                  |
| Type     | MESS                                                           |
| Contents | "Option number"<br>F-option_no.option<br>" has been modified." |

---

## Creating the Delete Functions

The delete functions are different for each data manager.

If you are developing the KSAM based application, read on below.

If you are developing the HP ALLBASE/SQL based applications, turn to HP ALLBASE/SQL Based Applications.

If you are developing the HP TurboIMAGE/iX based application, turn to HP TurboIMAGE/iX Based Applications.

### Creating the Delete Functions for KSAM Based Applications

You must create four functions to delete product and option records. The processing for this operation is a little different from earlier parts of the application. The delete functions require the user to verify that the selected option or product is the correct item to be deleted. You will use a query message to do this.

When the user selects a product for deletion, the application first checks to see if there are any options other than option number 000 associated with the product. If options do exist, the user is alerted and asked to confirm the deletion of the options.

The user must delete the options before the product can be deleted. When a product is deleted, option number 000 for the product is also deleted.

If option number 000 for a product does not exist when it is first accessed, the user is alerted and given the option of deleting the product.

#### Modifying the delete\_product function

##### To modify the function:

1. Modify the *delete\_product* function, using the logic commands shown below.
2. Regenerate the function.

**Function** - *delete\_product*.

## 10-48 Expanding Logic Blocks



```

1 DEFINE %PROD% .product
2 IF V-product_status = C-no_record THEN
    MESSAGE no_product ; EXIT
3 LINK 2 F-product_no%PROD% "000" F-option_key.option
4 FILE *READ option ; ENTER 17
5 VISIT del_all_options
6 IF 8 *ON THEN ENTER 15
7 MESSAGE delete_prod
8 IF V-confirm <> "Y" THEN ENTER 15
9 SERIES 3 3
10 FILE *DELETE option ; EXIT
11 FILE *DELETE product ; EXIT
12 MESSAGE prod_delete_ok
13 MOVE C-no_record V-product_status
14 EXIT
15 MESSAGE prod_delete_fail
16 EXIT
17 VISIT del_corrupt_prod
18 EXIT

```

The modified *delete\_product* function operates as follows:

- Line 1            This defines an identifier for the *product* file.
- Line 2            This step ensures that a product record is current.
- Line 3            This step links the current product number and the number *000* into the key field on the *option* file record buffer.
- Line 4            This step reads the option file for a record with the key value set up in step 3. If the record is not found, the file may be corrupted. Every product must have an option number 000. Control passes to step 17 if an error is detected.
- Line 5            The *del\_all\_options* function deletes all of the options for the current product. The *del\_all\_options* uses switch number 8 to indicate its exit status. If switch number 8 is *off* when the function exits, all options other than option

|                     |                                                                                                                                                                                                                                                                                     |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | number 000 for the current product have been deleted successfully.                                                                                                                                                                                                                  |
| Line 6              | This step tests the status of switch number 8. If this switch is <i>on</i> , the options for the product have not been deleted. Control passes to step 15 to display a message to the user.                                                                                         |
| Line 7              | The message <i>delete_prod</i> will be created soon. This message asks the user to confirm the deletion of the product. The user's response is returned to the variable <i>confirm</i> .                                                                                            |
| Line 8              | If the user's response to the query message is "Y", control passes to step 9. If the user's response is anything else, control passes to step 15.                                                                                                                                   |
| Line 9              | This step uses the SERIES command to execute step 3 again to rebuild the key for option number 000 for the current product. This is necessary since the <i>del_all_options</i> function may change the <i>option</i> file buffer contents.                                          |
| Line 10             | The record for option 000 is deleted.                                                                                                                                                                                                                                               |
| Line 11             | The product record is deleted.                                                                                                                                                                                                                                                      |
|                     | If a file error is detected in either of these steps, the function exits immediately. In this case, control returns to the <i>product</i> process, and the error condition processing is executed. You may want to look back at the <i>product</i> process to see how this is done. |
| Lines 12, 13 and 14 | These steps display a confirmation message for the user and set the <i>product_status</i> variable. The function then exits.                                                                                                                                                        |
| Lines 15 and 16     | Control passes to these steps if the user does not confirm the deletion of the product, or the options for the product are not deleted successfully. These steps display a message, and then exit.                                                                                  |
| Lines 17            | Control passes to this step if the first read of the <i>option</i> file does not succeed. This can only occur if option number 000 for the current product does not exist, indicating corruption of the <i>option</i> file or the <i>product</i> file. This step visits the         |

## 10-50 Expanding Logic Blocks

function *del\_corrupt\_prod* to delete the current product record, and any option records for the product.

Lines 18           The function exits.

3. Add following message, which is required by the *delete\_product* function:

**Message - *delete\_prod***

|               |                                                        |
|---------------|--------------------------------------------------------|
| Name          | delete_prod                                            |
| Type          | QUERY                                                  |
| Response Item | V-confirm                                              |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this product." |

**Modifying the *del\_all\_options* function**

The *del\_all\_options* function deletes all options for a product except option number 000. It is called from the *delete\_product* function.

The *del\_all\_options* function uses switch number 8 to indicate its exit status. If the options for a product are deleted successfully, switch number 8 is set *off* when the function exits. Otherwise, switch number 8 is set *on*.

**To modify the *del\_all\_options* function:**

1. Create the *del\_all\_options* function using the following commands.
2. Generate the function.

```
1 ON 8
2 FILE *NEXT option ; ENTER 10
3 IF F-product_no.option <> F-product_no.product
   THEN OFF 8 ; EXIT
4 MESSAGE delete_options
5 IF V-confirm <> "Y" THEN EXIT
6 SCROLL 6 F-option_no.option 4 F-description.option 8 "deleted"
7 FILE *DELETE option
8 FILE *NEXT option ; ENTER 10
9 IF F-product_no.option <> F-product_no.product THEN
   ENTER 12 ELSE ENTER 6
10 IF *IOSTATUS <> W-end_of_file THEN MESSAGE file_error ; EXIT
11 SCROLL 9 "***** All Options Deleted for Product # " & F-product_no.product" *****"
12 OFF 8
13 EXIT
```

The *delete\_all\_options* function operates as follows:

- Line 1            This step turns switch number 8 *on*. If the function deletes the option records successfully, this switch is set *off* before the function exits.
- Line 2            The FILE \*NEXT command reads the next record from the *option* file. Since this function is called after the FILE \*READ command in the *delete\_product* function, the FILE \*NEXT command attempts to retrieve the record for option number 001. If the FILE \*NEXT command encounters an error, control passes to step 10.
- Line 3            s step tests the value in the *product\_no* field on the record just read. If this value is not equal to the product number of the current product, the record just read is an option record for a different product. In this case, the current product does not have any options other than option number 000, so no options require deletion. In this case, switch number 8 is set *off* and the function exits.
- If the record just read is an option record for the current product (that is, the value in the *product\_no* field matches the current product number), the product must have at least one option other than option number 000.
- Line 4            This step displays a query message that asks the user to confirm the deletion of the options for this product.
- Line 5            If the user enters “Y” in response to the message, control passes to step 6. Otherwise, the function exits.
- Lines 6 to 9      These steps form a loop that deletes all the option records for the current product.
- This command displays a line of data containing six spaces, the current option number, four further spaces, the description of the option, eight further spaces, and the word “deleted”.
- Line 10           This step deletes the current *option* file record.

## 10-52 Expanding Logic Blocks

Line 11 This step reads the next record in the *option* file. If an error is encountered, control passes to step 10.

Line 12 If the value in the *product\_no* field in the record just read does not match the number for the current product, the record just read is an option record for a different product. This means all options for the current product have been deleted. In this case, control passes to step 11. Otherwise, control returns to step 6 to continue the option deletion loop.

The option deletion loop continues until there are no options for the current product left in the *option* data file. Control passes to step 12 when the last option record for the current product has been deleted.

Line 13 Control passes to this step if an error occurs in either of the FILE \*NEXT commands. This step tests the value in \*IOSTATUS. If this value is not equal to 19110 (the value of the constant *end\_of\_file*), indicating the FILE \*NEXT command encountered an end of file condition, control passes to step 12. An end of file condition may occur after the last record in the file has been deleted. This situation does not indicate an incorrect condition.

If the file error is caused by any condition other than encountering an end of file condition, then a message is displayed and the function exits.

Line 14 The user is informed that all of the options for the current product have been deleted.

Line 15 This step sets switch number 8 *off* to indicate that all options for the current product have been deleted successfully.

Line 16 The function exits.

3. The *delete\_all\_options* function requires one message. Create this message now.

**Message - *delete\_options***

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| Name          | delete_options                                                         |
| Type          | QUERY                                                                  |
| Response Item | V-confirm                                                              |
| Contents      | "This product has options. "<br>"Enter 'Y' to confirm their deletion." |

### Creating the del\_corrupt\_prod function

The *del\_corrupt\_prod* function is called from the *delete\_product* function if the first read of the *option* file fails. This can only occur if the record for option number 000 does not exist, indicating that the record for the product or option is corrupted.

The *del\_corrupt\_prod* function deletes the product and option records for a corrupted product record.

#### To create the del\_corrupt\_prod function:

1. Create this function as shown below.
2. Generate the function.

```

1 MESSAGE corrupt_prod_del
2 IF V-confirm <> "Y" THEN MESSAGE prod_delete_fail ; EXIT
3 FILE *FIND option *KEY= F-product_no.product ; ENTER 14
4 FILE *NEXT option ; ENTER 14
5 IF F-product_no.option <> F-product_no.product
   THEN ENTER 9
6 SCROLL 2 F-option_no.option 4 F-description.option
   8 "deleted"
7 FILE *DELETE option
8 ENTER 4
9 FILE *DELETE product ; ENTER 12
10 MOVE C-no_record V-product_status
11 EXIT
12 MESSAGE file_error
13 EXIT
14 IF *IOSTATUS = N-record_not_found | *IOSTATUS = N-end_of_file

```

### 10-54 Expanding Logic Blocks

THEN ENTER 9 ELSE MESSAGE file\_error ; EXIT

The *del\_corrupt\_prod* function operates as follows.

- Line 1            This step displays a query message asking the user to confirm the deletion of the product and options.
- Line 2            If the user enters “Y” in response to the message, control passes to step 3. Otherwise, the *prod\_delete\_fail* message is displayed and the function exits.
- Line 3            This step uses the FILE \*FIND command to search the file for a record with a key value equal to or greater than the key specified. In this case, the command uses the current product number as the key for the search. This command finds the first existing option record for the current product since the option key is a concatenation of the product number and the option number.
- If the FILE \*FIND command does not find a record, control passes to step 14.
- Line 4            The FILE \*NEXT command in this step retrieves the record into the file buffer. Control passes to step 14 if the record is not retrieved successfully. Steps 5 to 9 form a loop that deletes all the existing options for the current product.
- Line 5            This step tests the value in the *product\_no* field for the record just read. If the record just read is an option record for a different product, no option records exist for the current product or all option records for the product have been deleted. In this case, control passes to step 9 to delete the product record.
- Line 6            This step scrolls the current option details, then the word “deleted” in the scroll area of the current screen.
- Line 7            This step deletes the current record from the *option* file.
- Line 8            Control is passed to step 5 to continue the option deletion loop.
- Line 9            Control passes to this step after the last option record for the current product has been deleted. This step deletes the

**Expanding Logic Blocks 10-55**

current product record from the *product* file. If an error occurs during this operation, control passes to step 13.

- Line 10 This step sets the value of the variable *product\_status* to indicate that there is no current product record.
- Line 11 The function exits at this point.
- Line 12 Control passes to this step if a file access error is detected in step 10. This step displays the file error message.
- Line 13 The function exits.
- Line 14 Control passes to this step if a file access error is detected in step 4 or step 5. If the file error is caused by failure to find a record or reaching the end of the file, control passes to step 10 to delete the current product record. This situation can occur if the first FILE \*FIND command fails to find any options for the current product, or the FILE \*NEXT command in step 5 reaches the end of file after the last record in the file has been deleted.

If the file access has failed for any other reason, the *file\_error* message is displayed and the function exits.

3. You must now create the *corrupt\_prod\_del* message. This message is displayed by the *del\_corrupt\_prod* function.

**Message - *corrupt\_prod\_del***

|               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| Name          | corrupt_prod_del                                                               |
| Type          | QUERY                                                                          |
| Response Item | V-confirm                                                                      |
| Contents      | "Product/option records corrupted. "<br>"Enter 'Y' to confirm their deletion." |

This completes the items for the product deletion part of this application. Now the option deletion function must be created.

**Deleting Options**

**Function - *delete\_option***

**10-56 Expanding Logic Blocks**



The option deletion operation is more straightforward and should not require any explanation.

**To create the delete\_option function:**

1. Enter the following steps:

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option_warn ; EXIT
2 MESSAGE delete_option
3 IF V-confirm <> "Y" THEN MESSAGE opt_delete_fail ; EXIT
4 FILE *DELETE option
5 MESSAGE opt_delete_ok
6 MOVE C-no_record V-option_status
7 EXIT
```

2. Create the messages needed for this function.

a. **Message - delete\_option**

The *delete\_option* function requires the following messages:

|               |                                                       |
|---------------|-------------------------------------------------------|
| Name          | delete_option                                         |
| Type          | QUERY                                                 |
| Response Item | V-confirm                                             |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this option." |

b. **Message - opt\_delete\_ok**

|          |                            |
|----------|----------------------------|
| Name     | opt_delete_ok              |
| Type     | MESS                       |
| Contents | "Option has been deleted." |

c. **Message - opt\_delete\_fail**

|          |                                |
|----------|--------------------------------|
| Name     | opt_delete_fail                |
| Type     | WARN                           |
| Contents | "Option has NOT been deleted." |

The next section contains instructions for readers creating the HP ALLBASE/SQL based application. Turn to the Summary to continue with this training guide.

## Creating Functions for HP ALLBASE/SQL Based Applications

### Deleting a Product

The *delete\_product* function is an existing function that will be modified to check for the existence of option records other than option number 000 for the current product. If they exist, it displays a message requesting the user to confirm their deletion. The function then requests the user to confirm the deletion of the product. If the user chooses to delete the product, the function deletes the record for option 000 and deletes the product record.

The function is called from the *prod\_opt\_update* decision table.

This function calls three SQL logic blocks. They are *opt\_sel*, *opt\_del*, and *prod\_del*. The operation of the function and these SQL logic blocks is explained below.

This function also requires some new messages, and the *delete\_prod* message must be modified.

#### To modify the delete\_product function:

1. Modify and regenerate the function and the SQL logic blocks as shown below.

```
1 IF V-product_status = C-no_record THEN
    MESSAGE no_product ; EXIT
2 SQL opt_sel ; ENTER 15
3 FILE *NEXT option ; ENTER 14
4 MESSAGE delete_options
5 IF V-confirm <> "Y" THEN ENTER 16
6 SQL opt_del ; ENTER 15
7 MESSAGE all_opt_deleted
8 MESSAGE delete_prod
9 IF V-confirm <> "Y" THEN ENTER 16
```

### 10-58 Expanding Logic Blocks

```

10 SQL prod_del ; ENTER 15
11 MESSAGE prod_delete_ok
12 MOVE C-no_record V-product_status
13 EXIT
14 IF *IOSTATUS = N-end_of_file THEN ENTER 8
15 MESSAGE file_error
16 MESSAGE prod_delete_fail
17 EXIT

```

The *delete\_product* function operates as follows:

- Line 1            This step ensures that a product record is current.
- Line 2            This step calls the SQL logic block *opt\_sel*, which declares and opens a cursor for the *option* table. If an error occurs, control passes to step 15.
- Line 3            This step retrieves the first record for the cursor. Control passes to step 14 if an error is detected.
- Lines 4 and 5    If options do exist for the product, the user is requested to confirm their deletion. If the user confirms the deletion of the options, the function passes control to step 6. Otherwise, control passes to step 16.
- Lines 6 and 7    The *opt\_del* SQL logic block deletes all of the options for the current product, and the *ALL\_opt\_deleted* message is displayed.
- Line 8            The message *delete\_prod* will be added soon. This message asks the user to confirm the deletion of the product. The user's response is returned to the variable *confirm*.
- Line 9            If the user's response to the query message is "Y", control passes to step 10. If the user's response is anything else, control passes to step 16.
- Line 10           The product record is deleted when this function calls the *prod\_del* SQL logic block.
- Lines 11, 12 and 13    These steps display a confirmation message for the user and set the *product\_status* variable. The function then exits.

Line 14           Control passes to this step if no option records are found for the product. If HP ALLBASE/SQL reaches the start or the end of the file, the function passes control to step 8 to delete a product.

Lines 15 and     Control passes to these steps if the user does not confirm  
16               the deletion of the product, or the options or the product are not deleted successfully. These steps display a message.

Line 17           The function exits.

2. The *delete\_product* function requires you to add the following QUERY type message. Create this message now.

**Message - *delete\_prod***

|               |                                                        |
|---------------|--------------------------------------------------------|
| Name          | delete_prod                                            |
| Type          | QUERY                                                  |
| Response Item | V-confirm                                              |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this product." |

3. Now create the following messages:

a. **Message - *delete\_options***

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| Name          | delete_options                                                         |
| Type          | QUERY                                                                  |
| Response Item | V-confirm                                                              |
| Contents      | "This product has options. "<br>"Enter 'Y' to confirm their deletion." |

b. **Message - *all\_opt\_deleted***

|          |                                                                                      |
|----------|--------------------------------------------------------------------------------------|
| Name     | all_opt_deleted                                                                      |
| Type     | MESS                                                                                 |
| Contents | "Number of options deleted for product "<br>F-product_no.product<br>" is " *ROWCOUNT |

### **Modifying SQL Logic Blocks**

The *ALL\_opt\_deleted* message introduces a new communication area field. After any operation that modifies a table, HP ALLBASE/SQL returns a value

### **10-60 Expanding Logic Blocks**

to \*ROWCOUNT to indicate the number of rows that are affected by the command. In this example, the value in \*ROWCOUNT is the number of rows deleted from the *option* table.

**To modify the SQL logic blocks:**

1. Modify the blocks as indicated below:

a. **SQL Logic Block - *opt\_sel***

The SQL command in step 2 executes the SQL logic block *opt\_sel*. This SQL logic block contains the following command:

```
SELECT :option FROM sqlgrp.option
WHERE product_no = :F-product_no.product
AND option_no > :N-zero;
```

This SQL logic block declares and opens a cursor on the *option* table. The active set for this cursor is the set of records for the current product that have an option number that is strictly greater than zero. Note that this select command does not find option number zero for a product. If a product has no options other than the base option (number 000), the active set for this cursor is empty.

The FILE \*NEXT command in step 3 of the *delete\_product* function retrieves the first record for the cursor. If this command is successful, steps 4 to 7 of the function display a message, and then delete the options if the user enters “Y” in response to the message.

If the FILE \*NEXT command encounters an end-of-file condition without retrieving a record, the SELECT command has not found any option records for the product. In this case, control passes to step 14 and then to step 8, to allow the user to delete the product. If the file error is due to any condition other than encountering the end of file, control passes to step 15.

b. **SQL Logic Block - *opt\_del***

The SQL logic block called by step 6 of the function deletes records from the *option* table. The *opt\_del* SQL logic block contains the following command:

```
DELETE FROM sqlgrp.option
  WHERE product_no = :F-product_no.product
  AND option_no > 000;
```

This SQL block demonstrates the technique for deleting records without using a cursor. In this case, the SQL DELETE command uses an explicit search condition to specify the rows to be deleted.

c. **SQL Logic Block** *prod\_del*

Step 10 of the *delete\_product* function calls the SQL logic block *prod\_del*. This SQL logic block deletes the record for option number 000 from the *option* table, and then deletes the product record from the *product* table. The second part of this SQL logic block already exists.

Modify the *prod\_del* SQL logic block so that it contains the following commands:

```
DELETE FROM sqlgrp.option
  WHERE product_no = :F-product_no.product;
```

```
DELETE FROM sqlgrp.product
  WHERE product_no = :F-product_no.product;
```

This SQL logic block demonstrates the use of more than one command in an SQL logic block. An SQL logic block can contain up to eight SQL commands if it does not contain a SELECT command. Note that you must terminate each command with a semicolon.

## Deleting Options

The technique for deleting option records is much simpler than the method for deleting a product.

### To enter the *delete\_option* function:

The *delete\_option* function is called by the *prod\_opt\_update* decision table.

**Function** - *delete\_option*

## 10-62 Expanding Logic Blocks

The option deletion operation is more straightforward and should not require any explanation.

1. Enter the following steps:

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option_warn ; EXIT
2 MESSAGE delete_option
3 IF V-confirm <> "Y" THEN MESSAGE opt_delete_fail ; EXIT
4 FILE *DELETE option ; ENTER 8
5 MESSAGE opt_delete_ok
6 MOVE C-no_record V-option_status
7 EXIT
8 MESSAGE file_error
9 EXIT
```

This function uses a FILE \*DELETE command to delete a record from the *option* table. This command is equivalent to the following SQL command:

```
DELETE FROM sqlgrp.option
WHERE CURRENT OF option_key_sel;
```

The cursor is declared and opened by the SQL logic block *option\_key\_sel* called by the function *option\_key\_read*. Note that the cursor name is the same as the name of the SQL logic block that declares and opens the cursor.

2. Create the following messages:

- a. **Message** - *delete\_option*

|               |                                                       |
|---------------|-------------------------------------------------------|
| Name          | delete_option                                         |
| Type          | QUERY                                                 |
| Response Item | V-confirm                                             |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this option." |

- b. **Message** - *opt\_delete\_ok*

|      |               |
|------|---------------|
| Name | opt_delete_ok |
| Type | MESS          |

Contents "Option has been deleted."

c. **Message** - *opt\_delete\_fail*

Name opt\_delete\_fail  
Type WARN  
Contents "Option has NOT been deleted."

The next section contains instructions for readers creating the HP TurboIMAGE/iX based application. Turn to the Summary to continue with this training guide.

## Creating the Functions for HP TurboIMAGE/iX Based Applications

You must create four functions to delete product and option records. The processing for this operation is a little different from earlier parts of the application. The delete functions require the user to verify that the selected option or product is the correct item to be deleted. You will use a query message to do this.

When the user selects a product for deletion, the application first checks to see if there are any options other than option number 000 associated with the product. If options do exist, the user is alerted and asked to confirm the deletion of the options.

The user must delete the options before the product can be deleted. When a product is deleted, option number 000 for the product is also deleted.

If option number 000 for a product does not exist when it is first accessed, the user is alerted and given the option of deleting the product.

### To modify the delete\_product function:

1. Modify the *delete\_product* function so that it contains the following steps
2. Then regenerate the function.

```
1 DEFINE %PROD% .product
2 IF V-product_status = C-no_record THEN
  MESSAGE no_product ; EXIT
3 LINK 2 F-product_no%PROD% "000" F-option_key.option
```

## 10-64 Expanding Logic Blocks



```

4 DM IMAGE *LOCK :D-traindb :R-option
5 FILE *READ option *INDEX = product_no
  *KEY = S-product_no.product_scrn ; ENTER 18
6 VISIT del_all_options
7 IF 8 *ON THEN ENTER 16
8 MESSAGE delete_prod
9 IF V-confirm <> "Y" THEN ENTER 16
10 SERIES 5 5
11 FILE *DELETE option ; EXIT
12 FILE *DELETE product ; EXIT
13 MESSAGE prod_delete_ok
14 MOVE C-no_record V-product_status
15 EXIT
16 MESSAGE prod_delete_fail
17 EXIT
18 VISIT del_corrupt_prod
19 EXIT

```

The modified *delete\_product* function operates as follows:

|        |                                                                                                                                                                                                                                                                                                                                 |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This defines an identifier for the <i>product</i> file.                                                                                                                                                                                                                                                                         |
| Line 2 | This step ensures that a product record is current.                                                                                                                                                                                                                                                                             |
| Line 3 | This step links the current product number and the number <i>000</i> into the key field on the <i>option</i> file record buffer.                                                                                                                                                                                                |
| Line 4 | This step locks the option file before it is read.                                                                                                                                                                                                                                                                              |
| Line 5 | This step reads the option file for a record with the key value set up in step 3. If the record is not found, the file may be corrupted. Every product must have an option number 000. Control passes to step 18 if an error is detected.                                                                                       |
| Line 6 | The <i>del_all_options</i> function deletes all of the options for the current product. The <i>del_all_options</i> uses switch number 8 to indicate its exit status. If switch number 8 is <i>off</i> when the function exits, all options other than option number 000 for the current product have been deleted successfully. |

Line 7            This step tests the status of switch number 8. If this switch is *on*, the options for the product have not been deleted. Control passes to step 16 to display a message to the user.

Line 8            The message *delete\_prod* will be created soon. This message asks the user to confirm the deletion of the product. The user's response is returned to the variable *confirm*.

Line 9            If the user's response to the query message is "Y", control passes to step 10. If the user's response is anything else, control passes to step 16.

Line 10           This step uses the SERIES command to execute step 4 again to place a record in the file record buffer. This is necessary since the *del\_all\_options* function ends with no record in the file record buffer.

Line 11           The record for option 000 is deleted.

Line 12           The product record is deleted.

                  If a file error is detected in either of these steps, the function exits immediately. In this case, control returns to the *product* process, and the error condition processing is executed. You may want to look back at the *product* process to see how this is done.

Lines 13, 14      These steps display a confirmation message for the user and and 15            set the *product\_status* variable. The function then exits.

16 and 17         Control passes to these steps if the user does not confirm the deletion of the product, or the options for the product are not deleted successfully. These steps display a message, and then exit.

Line 18           Control passes to this step if the first read of the *option* file does not succeed. This can only occur if option number 000 for the current product does not exist, indicating corruption of the *option* file or the *product* file. This step visits the function *del\_corrupt\_prod* to delete the current product record, and any option records for the product.

Line 19           The function exits.

## 10-66 Expanding Logic Blocks

3. Add the following message, which is required by the *delete\_product* function:

**Message - *delete\_prod***

|               |                                                        |
|---------------|--------------------------------------------------------|
| Name          | delete_prod                                            |
| Type          | QUERY                                                  |
| Response Item | V-confirm                                              |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this product." |

**To modify the *del\_all\_options* function:**

The *del\_all\_options* function deletes all options for a product except option number 000. It is called from the *delete\_product* function.

The *del\_all\_options* function uses switch number 8 to indicate its exit status. If the options for a product are deleted successfully, switch number 8 is set *off* when the function exits. Otherwise, switch number 8 is set *on*.

The *del\_all\_options* function contains the following commands.

1. Create and generate this function now.

```
1 ON 8
2 FILE *NEXT option ; ENTER 10
3 MESSAGE delete_options
4 IF V-confirm <> "Y" THEN EXIT
5 DM IMAGE *LOCK :D-traindb :R-option
6 IF F-option_no.option <> "000" THEN FILE *DELETE option
7 SCROLL 6 F-option_no.option 4 F-description.option 8 "deleted"
8 FILE *NEXT option ; ENTER 10
9 ENTER 6
10 IF *IOSTATUS <> N-end_of_file & *IOSTATUS <> N-end_of_chain
    THEN MESSAGE file_error ; EXIT
11 SCROLL 9 "***** All Options Deleted for Product # "
    F-product_no.product "*****"
12 OFF 8
13 EXIT
```

The *delete\_all\_options* function operates as follows:

- Line 1            This step turns switch number 8 *on*. If the function deletes the option records successfully, this switch is set *off* before the function exits.
- Line 2            The FILE \*NEXT command reads the next record from the *option* file. Since this function is called after the FILE \*READ command in the *delete\_product* function, the FILE \*NEXT command attempts to retrieve the record for option number 001. If the FILE \*NEXT command encounters an error, control passes to step 10.
- Line 3            This step displays a query message that asks the user to confirm the deletion of the options for this product.
- Line 4            If the user enters “Y” in response to the message, control passes to step 5. Otherwise, the function exits.
- Line 5            This step places an HP TurboIMAGE/iX logical lock on the *option* data set. This lock is released by the *product\_proc* process.
- Line 6            Steps 7 to 9 form a loop that deletes all the option records for the current product.
- This command tests to ensure that the option number does not equal “000”. If it does not equal “000”, then the option is deleted.
- Line 7            This step displays a line of data containing six spaces, the current option number, four further spaces, the description of the option, eight further spaces, and the word “deleted”.
- Line 8            This step reads the next record in the *option* file. If an error is encountered, control passes to step 10.
- Line 9            If no errors are encountered in the previous step, there are more option records for this product. Control returns to step 6 to continue the option deletion loop.
- The option deletion loop continues until there are no options for the current product left in the *option* data file. Control passes to step 10 when the last option record for the current product has been deleted.

## 10-68 Expanding Logic Blocks

- Line 10           Control passes to this step if an error occurs in the FILE \*NEXT command. This step tests the value in \*IOSTATUS. If this value is not equal to 19110 (the value of the constant *end\_of\_file*) or 19115 (the value of the constant *end\_of\_chain*), control passes to step 11. An end of file condition may occur after the last record in the file has been deleted, and an end of chain condition may occur after the last record in a chain of linked detail data set records has been deleted. This situation does not indicate an incorrect condition.
- If the file error is caused by any other condition, then a message is displayed and the function exits.
- Line 11           The user is informed that all of the options for the current product have been deleted.
- Line 12           This step sets switch number 8 *off* to indicate that all options for the current product have been deleted successfully.
- Line 13           The function exits.

2. This function requires one message. Create this message.

**Message - *delete\_options***

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| Name          | delete_options                                                         |
| Type          | QUERY                                                                  |
| Response Item | V-confirm                                                              |
| Contents      | "This product has options. "<br>"Enter 'Y' to confirm their deletion." |

**To create the *del\_corrupt\_prod* function:**

The *del\_corrupt\_prod* function is called from the *delete\_product* function if the first read of the *option* file fails. This can only occur if the record for option number 000 does not exist, indicating that the record for the product or option is corrupted.

The *del\_corrupt\_prod* function deletes the product and option records for a corrupted product record.

The *del\_corrupt\_prod* function contains the following commands.

1. Create and generate the function as shown below.

```
1 MESSAGE corrupt_prod_del
2 IF V-confirm <> "Y" THEN MESSAGE prod_delete_fail ; EXIT
3 DM IMAGE *LOCK :D-traindb :R-option
4 FILE *FIND option *KEY= F-product_no.product ; ENTER 15
5 FILE *NEXT option ; ENTER 15
6 IF F-product_no.option <> F-product_no.product
   THEN ENTER 10
7 SCROLL 2 F-option_no.option 4 F-description.option
   8 "deleted"
8 FILE *DELETE option
9 ENTER 5
10 FILE *DELETE product ; ENTER 13
11 MOVE C-no_record V-product_status
12 EXIT
13 MESSAGE file_error
14 EXIT
15 IF *IOSTATUS = N-record_not_found | *IOSTATUS = N-end_of_file
   THEN ENTER 10 ELSE MESSAGE file_error ; EXIT
```

The *del\_corrupt\_prod* function operates as follows.

- |        |                                                                                                                                                                               |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Line 1 | This step displays a query message asking the user to confirm the deletion of the product and options.                                                                        |
| Line 2 | If the user enters “Y” in response to the message, control passes to step 3. Otherwise, the <i>prod_delete_fail</i> message is displayed and the function exits.              |
| Line 3 | This step places an HP TurboIMAGE/iX logical lock on the <i>option</i> file. This lock is released by the <i>product_proc</i> process.                                        |
| Line 4 | This step uses the FILE *FIND command to search the file for a record with a key value equal to or greater than the key specified. In this case, the command uses the current |

## 10-70 Expanding Logic Blocks

product number as the key for the search. This command finds the first existing option record for the current product since the option key is a concatenation of the product number and the option number.

If the FILE \*FIND command does not find a record, control passes to step 15.

- Line 5 The FILE \*NEXT command in this step retrieves the record into the file buffer. Control passes to step 15 if the record is not retrieved successfully. Steps 5 to 9 form a loop that deletes all the existing options for the current product.
- Line 6 This step tests the value in the *product\_no* field for the record just read. If the record just read is an option record for a different product, no option records exist for the current product or all option records for the product have been deleted. In this case, control passes to step 9 to delete the product record.
- Line 7 This step scrolls the current option details, then the word “deleted” in the scroll area of the current screen.
- Line 8 This step deletes the current record from the *option* file.
- Line 9 Control is passed to step 5 to continue the option deletion loop.
- Line 10 Control passes to this step after the last option record for the current product has been deleted. This step deletes the current product record from the *product* file. If an error occurs during this operation, control passes to step 13.
- Line 11 This step sets the value of the variable *product\_status* to indicate that there is no current product record.
- Line 12 The function exits at this point.
- Line 13 Control passes to this step if a file access error is detected in step 10. This step displays the file error message.
- Line 14 The function exits.
- Line 15 Control passes to this step if a file access error is detected in step 4 or step 5. If the file error is caused by failure to find

## Expanding Logic Blocks 10-71

a record or reaching the end of the file, control passes to step 10 to delete the current product record. This situation can occur if the first FILE \*FIND command fails to find any options for the current product, or the FILE \*NEXT command in step 5 reaches the end of file after the last record in the file has been deleted.

If the file access has failed for any other reason, the *file\_error* message is displayed and the function exits.

2. Create the *corrupt\_prod\_del* message. This message is displayed by the *del\_corrupt\_prod* function.

**Message** - *corrupt\_prod\_del*

|               |                                                                                |
|---------------|--------------------------------------------------------------------------------|
| Name          | corrupt_prod_del                                                               |
| Type          | QUERY                                                                          |
| Response Item | V-confirm                                                                      |
| Contents      | "Product/option records corrupted. "<br>"Enter 'Y' to confirm their deletion." |

This completes the items for the product deletion part of this application. Now the option deletion function must be created.

## Deleting Options

**Function** - *delete\_option*

The option deletion operation is more straightforward and should not require any explanation.

1. Create the *delete\_option* function. It contains the following steps:

```
1 IF V-option_status = C-no_record THEN
    MESSAGE no_option_warn ; EXIT
2 MESSAGE delete_option
3 IF V-confirm <> "Y" THEN MESSAGE opt_delete_fail ; EXIT
4 FILE *DELETE option
5 MESSAGE opt_delete_ok
6 MOVE C-no_record V-option_status
7 EXIT
```

## 10-72 Expanding Logic Blocks



2. Create the *delete\_option* function messages as shown below.

a. **Message** - *delete\_option*

|               |                                                       |
|---------------|-------------------------------------------------------|
| Name          | delete_option                                         |
| Type          | QUERY                                                 |
| Response Item | V-confirm                                             |
| Contents      | "Enter 'Y' to confirm "<br>"deletion of this option." |

b. **Message** - *opt\_delete\_ok*

|          |                            |
|----------|----------------------------|
| Name     | opt_delete_ok              |
| Type     | MESS                       |
| Contents | "Option has been deleted." |

c. **Message** - *opt\_delete\_fail*

|          |                                |
|----------|--------------------------------|
| Name     | opt_delete_fail                |
| Type     | WARN                           |
| Contents | "Option has NOT been deleted." |

---

## Summary

In this lesson you created the functions to add, modify and delete records in the product and option file.

You may now run your application. You will be able to add, modify, and delete product and option records. Test the `Scroll Options` function key too. If you have any problems, refer back to Handling Problems in chapter 5.

You can also complete the Self-Test Questions following the summary.

---

## Self Test Questions

**Question 1.** What does the TRANSACT command do, and what are the four arguments that you can use with this command?

**Question 2.** What steps are involved in producing a decision table?

**Question 3.** Which of the following items require generation?

- Validation tables.
- Help screens.
- Windows.
- Decision tables.

**Question 4.** Each screen field can only call one function. How can you make this function execute different steps according to whether the user has entered data in the field or not?

---

## Answers

**Answer 1.** The TRANSACT command allows you to define logically connected groups of file modification commands that constitute one transaction.

The three arguments to this command are \*BEGIN, \*END, \*UNDO and \*MEMO. TRANSACT \*BEGIN and TRANSACT \*END define the beginning and the end of the transaction. TRANSACT \*UNDO reverses all KSAM file transactions occurring since the TRANSACT \*BEGIN command, and undoes any uncommitted HP ALLBASE/SQL transactions. TRANSACT \*MEMO is used to include comments in an HP TurboIMAGE/iX log record, and has no effect on KSAM or HP ALLBASE/SQL transactions.

**Answer 2.** The following steps are involved in producing a decision table.

1. Defining the decision table header.
2. Defining the decision table questions.
3. Defining the decision table actions.
4. Defining the decision table relationships.
5. Generating the decision table.

**Answer 3.** Decision tables and windows must be generated. Validation tables and help screens do not need to be generated.

**Answer 4.** You can use an IF THEN ELSE command to perform different steps of a function according to the value of the \*ENTERED switch. If the switch is on, the user has entered data in the field. If the switch is off, the field has not been committed.

FINAL TRIM SIZE : 7.0 in x 8.5 in

## Developing Greater Reporting Capabilities

---

This is the final chapter of the Developer Self-Paced Training Guide. The chapter consists of two lessons. In the first lesson you will create a report that uses data from two files. In the second lesson you will create a calculated item that is used by the report. A calculated item is a special type of variable.

---

### Lesson 27-Learning More Reporting Techniques

---

#### Objectives

When you have completed this lesson, you will have learned how to create a report that uses data from both the product file and the option file. In developing this report, you will learn how to use the following reporting features.

- A start of report function.
- Sort field definitions and control breaks.
- Automatic totalling.
- Record selection criteria (HP TurboIMAGE/iX and KSAM only).
- File linkages and a post link read function (HP TurboIMAGE/iX and KSAM only).
- A before print function associated with a report line group.

HP ALLBASE/SQL developers will also create a select list.

You will also use a calculated item in this report. A calculated item is a special type of variable that is evaluated by executing a function every time the item is referenced.

These facilities allow you to develop reports that are considerably more flexible and powerful than the simple report in the earlier part of this training course.

---

## Creating The Product/Option Report

The product/option report that you are about to develop uses a start of report function to display a screen. This screen allows the user to enter the starting and finishing values for a range of product numbers for the report.

The report then lists each of these products and the options for each product. The report calculates the cost of stock for each option by multiplying the quantity on hand by the cost price. When all options for a product have been listed, the total cost of stock for that product is calculated and printed. Finally, the report calculates the total cost of stock for all products listed.

### Some Preparations

If you are developing the HP ALLBASE/SQL based application, read on below to create a select list that you will need to join the option table and the product table.

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, skip to Report Header to begin defining this report.

### Creating the HP ALLBASE/SQL Based Applications Select List

#### To create the select list:

The select list you will create, *prod\_opt*, requires an additional field specification *opt\_descript*. This field specification is identical to the existing *description* field specification.

1. Use the copying utility to create it now.
2. Then create and generate the *prod\_opt* select list.

## 11-2 Developing Greater Reporting Capabilities

The *prod\_opt* select list is as follows:

```
product_no = product_no,  
description = description,  
supplier_no,  
lead_time,  
option_no,  
opt_descript = option.description,  
cost,  
quantity,  
unit_measure
```

This select list uses the new field specification *opt\_descript*. This is necessary to avoid duplicate field specification names in the select list.

This select list requires you to enter the right hand side of the *product\_no*, *description*, and *opt\_descript* columns. These entries are required to define the columns completely to avoid ambiguity.

This select list also demonstrates the use of qualified names in the column definitions. The column definitions for the *product\_no* column and both *description* columns contain the name of the table as a qualifier. This qualifier is necessary to avoid ambiguous column definitions. The terms on the right hand side of these entries use the standard SQL syntax for referencing table columns. That is, you must reference a column as *table\_name.column\_name*.

Now you may begin defining the report.

## Defining the Report Header

### To define the report header:

1. Call up the report header screen and enter the data shown below.

Two sets of data are shown: one for KSAM and HP TurboIMAGE/iX and the other for HP ALLBASE/SQL.

2. Accept the default entries for any fields not listed here.
3. Press the **Commit Data** function key when you have completed the screen.

## KSAM Based and HP TurboIMAGE/iX Based Applications

|                        |               |
|------------------------|---------------|
| Report Name            | prod_opt      |
| Report File Designator | prodop        |
| Primary File[.Record]  | product       |
| Index                  | 1             |
| Characters per line    | 100           |
| Start of Report        | product_range |
| Function Name          |               |

The last field is one that KSAM developers and HP TurboIMAGE/iX developers haven't used yet. It is the name of a function that is executed when the report starts. In this application, the function displays a screen that asks the user for the range of product numbers to be included in the report. The report is produced when this function finishes.

## HP ALLBASE/SQL Based Applications

|                        |               |
|------------------------|---------------|
| Report Name            | prod_opt      |
| Report File Designator | prodop        |
| Primary File[.Record]  | prod_opt      |
| Characters per line    | 100           |
| Start of Report        | product_range |
| Function Name          |               |

This report uses an HP ALLBASE/SQL select list as the primary report file. The select list contains columns from two HP ALLBASE/SQL tables (the *product* table and the *option* table). While you develop this report, you will see how you can use the selection and sorting facilities provided by HP ALLBASE/SQL in conjunction with HP ALLBASE/4GL.

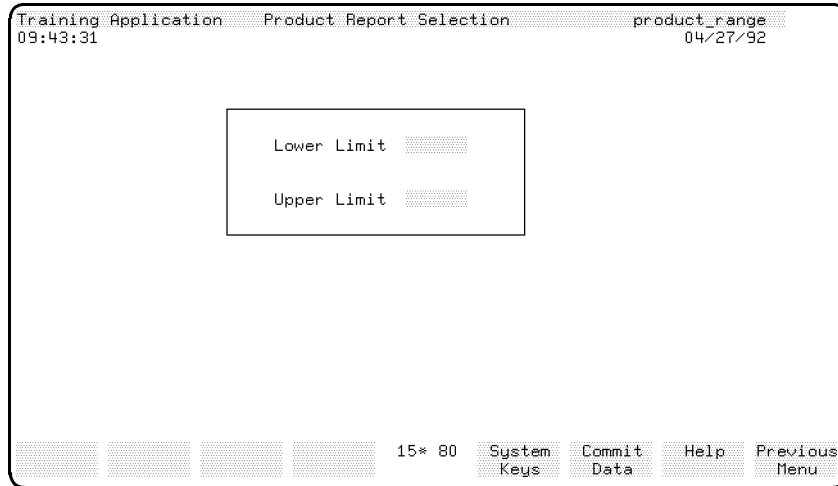
The last field contains the name of a function that is executed when the report starts. In this application, the function displays a screen that asks the user for the range of product numbers to be included in the report. The report is produced when this function finishes.

## 11-4 Developing Greater Reporting Capabilities



## Creating the Screen

Now you are ready to create the *product\_range* screen.



The screenshot shows a terminal window titled "Training Application" with a sub-header "Product Report Selection" and "product\_range". The time is 09:43:31 and the date is 04/27/92. The main content area contains two input fields: "Lower Limit" and "Upper Limit", each followed by a series of asterisks representing a text input field. At the bottom, there is a status bar with "15\* 80" and four menu items: "System Keys", "Commit Data", "Help", and "Previous Menu".

**product\_range Screen**

This data screen requires two data input fields. Neither field uses any data movement or logic.

### To create the screen:

1. Define the screen header
2. Paint the image.
3. Create both fields using the dictionary field specification *product\_no* to ensure that automatic product number range checking is done. You don't need to enter any screen field details.
4. Remember to generate the screen.

## Creating the Function

Next you create the *product\_range* function.

### Function - *product\_range*

This function is called as the start of report function for the *prod\_opt* report.

If you are developing the KSAM based or HP TurboIMAGE/iX based applications, continue reading below.

If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications for a description of the function you will create.

### KSAM Based and HP TurboIMAGE/iX Based Applications

1. Create the *product\_range* function.

It executes the *product\_range* screen, retrieves data from the screen, and initializes some items for use later in the report.

2. After creating it, generate the function.

```
1 SCREEN product_range
2 IF *S01 $>=$ *S02 THEN MOVELOOP 2 *S01 1
  *P02 -1 ELSE ENTER 5
3 MOVELOOP 2 *P01 1 *S01 1
4 SHOW
5 MOVE " " *P03
6 OFF *ALL
7 EXIT
```

The function operates as follows.

Line 1            This step executes the *product\_range* data screen. This screen allows the user to enter the range of products to be included in the report. When the user commits the screen, this step terminates and the next step is executed.

The *start of report* function is the only function in a report that can display a screen.

## 11-6 Developing Greater Reporting Capabilities

- Line 2            This step checks whether the lower limit value is less than the upper limit value. If it is, control passes to step 5.
- If the lower limit value is greater than the upper limit value, this step swaps the two values. The MOVELOOP command moves the two screen fields to two scratch-pad fields.
- Scratch-pad fields are general purpose working variables that assume the attributes and value of the data moved into them. If the source is numeric, the scratch-pad field becomes numeric. If the source is alphanumeric, the scratch-pad field is treated as an alphanumeric variable. You can reference scratch-pad fields by number in the form \*P01 to \*P99.
- You can also assign names to scratch-pad fields using the *Scratch-Pad Fields* screen in the dictionary menu. Refer to the *HP ALLBASE/4GL Developer Reference Manual* for more details on naming and accessing scratch-pad fields.
- In this case the MOVELOOP command is performed twice. The first screen field, \*S01, is moved to the second scratch-pad field \*P02. Then the second screen field, \*S02, is moved to the first scratch-pad field \*P01. Note that the step factor for the screen fields is positive, while the step factor for the scratch-pad fields is negative.
- Line 3            This step moves the contents of the scratch-pad fields back to the screen fields in reverse order. This swaps both the values. Note that both step factors are positive.
- Line 4            This step redisplay the screen field buffers. Step 3 moves values directly into the internal screen field buffers so this step displays the new values. The swap is complete and the user has seen the change on the screen.
- Lines 5, 6 and 7    These steps initialize scratch-pad field \*P03 with a single space and initialize all the user switches for later use in the report. The function then exits.

Now turn to Report Sorting to continue defining the report.

## HP ALLBASE/SQL Based Applications

The *product\_range* function requires two variables, *first\_product* and *last\_product*. Both variables have a length of six characters and have a type U edit code.

1. Create the variables for HP ALLBASE/SQL applications.
2. When finished, generate the function

The *product\_range* function is called as the start of report function for the *prod\_opt* report. It executes the *product\_range* screen, retrieves data from the screen, initializes some items for use later in the report, and declares and opens a cursor for the report.

```
1 SCREEN product_range
2 IF *S01 >= *S02 THEN MOVELOOP 2 *S01 1 *P02 -1
   ELSE ENTER 5
3 MOVELOOP 2 *P01 1 *S01 1
4 SHOW
5 MOVE *S01 V-first_product
6 MOVE *S02 V-last_product
7 SQL get_prod_opt
8 EXIT
```

The function operates as follows.

Line 1            This step executes the *product\_range* data screen. This screen allows the user to enter the range of products to be included in the report. When the user commits the screen, this step terminates and the next step is executed.

The *start of report* function is the only function in a report that can display a screen.

Line 2            This step checks whether the lower limit value is less than the upper limit value. If it is, control passes to step 5.

If the lower limit value is greater than the upper limit value, this step swaps the two values. The MOVELOOP command moves the two screen fields to two scratch-pad fields.

Scratch-pad fields are general purpose working variables that assume the attributes and value of the data moved into them. If the source is numeric, the scratch-pad field becomes numeric.

## 11-8 Developing Greater Reporting Capabilities

If the source is alphanumeric, the scratch-pad field is treated as an alphanumeric variable. You can reference scratch-pad fields by number in the form \*P01 to \*P99. You can also assign names to scratch-pad fields using the *Scratch-Pad Fields* screen in the dictionary menu. Refer to the *HP ALLBASE/4GL Developer Reference Manual* for more details on naming and accessing scratch-pad fields.

In this case the MOVELOOP command is performed twice. The first screen field, \*S01, is moved to the second scratch-pad field \*P02. Then the second screen field, \*S02, is moved to the first scratch-pad field \*P01. Note that the step factor for the screen fields is positive, while the step factor for the scratch-pad fields is negative.

- Line 3      This step moves the contents of the scratch-pad fields back to the screen fields in reverse order. This swaps both the values. Note that both step factors are positive.
- Line 4      This step redisplay the screen field buffers. Step 3 moves values directly into the internal screen field buffers so this step displays the new values. The swap is complete and the user has seen the change on the screen.
- Lines 5 and 6      These two steps move the first and last product numbers for the report to the variables *first\_product* and *last\_product*.
- Line 7      This step calls the SQL logic block *get\_prod\_opt* which declares and opens a cursor on the *prod\_opt* select list. You will create the logic block soon.
- Line 8      The function then exits.

Note that this function does **not** contain a FILE \*NEXT command. The report generator issues the equivalent of a FILE \*FIRST command to retrieve the first record in the active set for the cursor.

### SQL Logic Block - *get\_prod\_opt*.

The *get\_prod\_opt* SQL logic block contains the following SELECT command. Create and generate this SQL logic block now.

```
SELECT :prod_opt FROM product, option
WHERE product.product_no = option.product_no
AND product.product_no BETWEEN :V-first_product
AND :V-last_product
ORDER BY product_no, option_no;
```

To join two tables, HP ALLBASE/SQL requires that they have at least one common column. In this case, both the *product* table and the *option* table contain a *product\_no* column.

This SELECT command declares and opens a cursor on the *prod\_opt* select list. The active set for this cursor contains all the records where the value in the *product\_no* column is equal to or greater than the value in the variable *first\_product* and equal to or less than the value in the variable *last\_product*. For each product number in the specified range of product numbers, this select command retrieves the corresponding product records, and all option records for these products.

The select command also contains an ORDER BY clause to specify that the select list records are sorted in product number order, and within each product number, records are sorted by option number.

---

## Defining Report Sorting Levels

The report sorting levels determine the order in which the report items are listed, and also define control break levels.

Control breaks allow you to perform a variety of sub-heading and sub-totalling tasks. A control break for a given sort level occurs each time the value in a sort field value changes from one record to the next. This report uses one control break to print the subtotal line after each product.

### Menu Path

To access the report sorting screen:

1. From the main menu, select the *Reports* option.

### 11-10 Developing Greater Reporting Capabilities

2. Choose *Sorting*.

**Screen Description**

This screen allows you to define up to eight sort fields, giving you a total of eight control break levels. The sort fields must be data fields on the primary file that you specified in the report header. For each sort field, you must state whether the sort is to be in ascending or descending order. You can also specify whether the file is already in sequence.

| Developer                    |               | Report Sorting                      |                         | report_sort |             |             |      |               |
|------------------------------|---------------|-------------------------------------|-------------------------|-------------|-------------|-------------|------|---------------|
| Report name                  | prod_opt      | Secured                             | N                       | (Y/N)       |             |             |      |               |
|                              | Field Name    | ORDER                               | Ascending or Descending |             |             |             |      |               |
| 1st                          | product_no    | A                                   | (A/D)                   |             |             |             |      |               |
| 2nd                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 3rd                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 4th                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 5th                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 6th                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 7th                          |               |                                     | (A/D)                   |             |             |             |      |               |
| 8th                          |               |                                     | (A/D)                   |             |             |             |      |               |
| Is File already in sequence? |               | <input checked="" type="checkbox"/> | (Y/N)                   |             |             |             |      |               |
| Header                       | Record Select | Line Header                         | Generate Report         | 11* 63      | System Keys | Commit Data | Help | Previous Menu |

**Report Sorting Screen**

**To enter the field values:**

| <b>Field</b>       | <b>Entry</b> | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Report Name</b> | prod_opt     | This is the name of the report that you are defining.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Sort Fields</b> |              | <p>The next 16 fields define the eight sort fields and their sorting order. The sort field numbers <i>1st</i> through to <i>8th</i> specify the control break level and relate directly to subheading line groups <i>H1 - H8</i> and subtotal <i>T1 - T8</i> line groups.</p> <p>The sorting order is either ascending or descending. The collating sequence is listed in the <i>HP ALLBASE/4GL Developer Reference Manual</i>.</p> <p>This report only uses the <i>1st</i> sort field.</p> |

**11-12 Developing Greater Reporting Capabilities**



| Field                        | Entry                                        | Explanation                                                                                                                                                                                                                          |
|------------------------------|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1st Field Name               | product_no                                   | This is the name of the field on the primary data file which forms the report's first sort level and control break.<br><br>For this report, a control break occurs every time a product record is read from the primary report file. |
| ORDER                        | A                                            | This specifies that the records are to be sorted in ascending order.<br><br>Leave the remaining <i>Field Name</i> and <i>ORDER</i> fields blank.                                                                                     |
| Is File already in sequence? | Enter Y for KSAM or SQL, N for TurboImage/ix | This indicates that the file is already in the order when it is read by the key specified on the header screen. In this application, no sorting is done, but the sort fields are used to initiate control breaks.                    |

**To complete the screen:**

1. Press the **Commit Data** function key to create the sorting specifications and control break specifications for this report.

---

## Defining Report Selection Criteria

If you are developing the HP ALLBASE/SQL based application, you can turn to Line Headers. You do not require any record selection criteria because this is performed by the SELECT command in the *get\_prod\_opt* SQL logic block.

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, read on.

In the *product\_range* function, the user is requested to enter the range of products to be included in the report. This screen allows you to specify the fields that are tested to determine if a record is to be selected for reporting.

## Menu Path

### To access the selection criteria screen:

1. From the main menu, select the *Reports* option.
2. Choose *Record Selection*.

### Screen Description

The screen allows you to define up to 99 sets of criteria that must be met for a record to be included on the report. You can specify selections for records from the primary report file or any link files used in the report.

|                            |                    |                  |                           |
|----------------------------|--------------------|------------------|---------------------------|
| Developer                  | Selection Criteria |                  | report_select             |
| Report Name                | prod_opt           | Secured          | N (Y/N)                   |
| After Selection Function   |                    |                  |                           |
| Selection Number           | 1                  | Action           | A (A/C/I/D)               |
| Link                       | Link File[.record] | Field Spec. Name | product_no                |
| Selection Criteria: Values | FROM *S01          |                  |                           |
|                            | TO *S02            |                  |                           |
| -----                      |                    |                  |                           |
| Number                     | Link               | Field            | 'FROM' selection criteria |
| Header                     | Sorting            | Line Header      | Generate Report           |
| 3*                         | 30                 | System Keys      | Commit Data               |
|                            |                    | Help             | Previous Menu             |

**Selection Criteria Screen**

## 11-14 Developing Greater Reporting Capabilities

**To enter the field values:**

| <b>Field</b>                    | <b>Entry</b>            | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Report Name</b>              | prod_opt                | This is the name of the report that you are defining.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>After Selection Function</b> | Leave this field blank. | This is the name of the function that is to be executed after each record is read.<br><br>HP ALLBASE/4GL sets the switch *BYPASS <i>on</i> if the current record read from the file fails selection, or <i>off</i> if the record passes selection. You can use this function to perform your own manual selection by changing the status of the *BYPASS switch. You can use this screen to specify just an after selection function and carry out record selection entirely from within this function. |
| <b>Selection Number</b>         | 1                       | This number identifies this set of selection criteria.                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Action</b>                   | Accept the default.     | This is the standard action code indicating add, change, insert, or delete.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Link</b>                     | Leave this field blank. | If the selection criteria is to be applied to a link file, you must specify the link number in this field. You will learn about defining links to other files soon.<br><br>When you leave this field blank, the link file displayed in the next field is the primary data file <i>product</i> . By default this set of selection criteria is applied to the primary report file.                                                                                                                       |

| Field           | Entry      | Explanation                                                                                                                                                                                                                                                                                              |
|-----------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field Spec.Name | product_no | This is the name of the field that is tested by this set of selection criteria. The field must exist on the link file displayed in the previous field.                                                                                                                                                   |
| Values FROM     | *S01       | This is the lower limit value of the field. In this case it is the first screen field of the <i>product_range</i> data screen. The entry can be a number, a literal, a validation table, or range. The <i>HP ALLBASE/4GL Developer Reference Manual</i> lists the possible field types that may be used. |
| Values TO       | *S02       | This is the upper limit value of the field. In this case it is the second screen field of the <i>product_range</i> data screen. The range specified by the upper and lower limits includes the <i>FROM</i> and <i>TO</i> values.                                                                         |

**To complete the screen:**

1. Press the **Commit Data** function key to create the selection criteria.

---

## Line Headers

To print the details read from link files, HP ALLBASE/4GL uses the link detail line groups D2 to D9. For the KSAM based and the HP TurboIMAGE/iX applications, line group D2 is used for details from the *option* file.

For the HP ALLBASE/SQL based application, no file linkages are used, so the only detail line group used is D1.

This is the major difference between the KSAM and HP TurboIMAGE/iX line header information and the HP ALLBASE/SQL line header information.

If you are developing the KSAM based application or the HP TurboIMAGE/iX based application, read on below.

### 11-16 Developing Greater Reporting Capabilities

If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications.

### KSAM Based and HP TurboIMAGE/iX Based Applications

1. Press the **Line Header** function key to start defining the line headers for the report.

Only those fields which require specific entries are shown here.

2. Accept the default entry for any fields that are not listed here.

| Line Group | Line Number | Skip Lines Before Print | Skip Lines After Print |
|------------|-------------|-------------------------|------------------------|
| P1         | 01          | 0                       | 1                      |
|            | 02          | 0                       | 1                      |
| D1         | 01          | 2                       | 1                      |
|            | 02          | 1                       | 1                      |
|            | 03          | 1                       | 2                      |
| T1         | 01          | 2                       | 1                      |
| TF         | 01          | 3                       | 1                      |
| E1         | 01          | 0                       | 1                      |

3. Also define a header for the following line:

**D2.01** This is the first line of the second detail group. This line contains the option details and is printed as a result of a link to the *option* file. You will learn more about file linkages shortly.

```

Line Group      D2
Line Number     01
Action          A
Before Print Function print_option
  
```

The before print function *print\_option* tests to see if any options were found and it prints a message if none were found. You will create the function and message soon.

The E1.01 line is printed from the *print\_option* function mentioned above. It tells the user that no options were found for the particular product just printed.

4. Turn to Report File Linkages to continue developing the application.

### HP ALLBASE/SQL Based Applications

1. Press the **Line Header** function key to start defining the line headers for the report.

Only those fields which require specific entries are shown here.

2. Accept the default entry for any fields that are not listed here.

| Line Group | Line Number | Skip Lines Before Print | Skip Lines After Print |
|------------|-------------|-------------------------|------------------------|
| P1         | 01          | 0                       | 1                      |
|            | 02          | 0                       | 1                      |
| H1         | 01          | 2                       | 1                      |
|            | 02          | 1                       | 1                      |
|            | 03          | 1                       | 2                      |
| D1         | 01          | 0                       | 1                      |
| T1         | 01          | 2                       | 1                      |
| TF         | 01          | 3                       | 1                      |

You have now completed all the line headers required for this report.

### 11-18 Developing Greater Reporting Capabilities

---

## Report File Linkages

If you are developing the HP ALLBASE/SQL based application, you can turn to Painting the Report. You do not require any report file linkages because the *prod\_opt* select list performs this file linkage.

Otherwise, read on below.

This report uses a file linkage to read records from the *option* file whenever a product record is printed. After the product details have been printed, the link is completed by printing all the matching records from the *option* file.

### Menu Path

To access the file linkage screen:

1. From the main menu, select the *Reports* option.
2. Choose *File Linkages*.

### Screen Description

This screen allows you to define a link to another file from a particular line group. Each time the specified line group is printed the link is performed. You can define up to 999 links for a report.

| Developer                  |                      | File Linkages    |                 |                        |                | report_link        |                |               |
|----------------------------|----------------------|------------------|-----------------|------------------------|----------------|--------------------|----------------|---------------|
| Report Name                | prod_opt             |                  |                 |                        |                | Secured            | N (Y/N)        |               |
| Link Number                | 1                    |                  |                 |                        |                |                    |                |               |
| Action                     | A                    | (A/C/I/D)        |                 |                        |                |                    |                |               |
| Link From Line Group       | D1                   | Print Line Group | D2              | End of Link Line Group |                | Post-Read Function | option_present |               |
| Link File[.Record]         | option               |                  |                 |                        |                | Index Number       | 2              |               |
| *KEY=                      | F-product_no,product |                  |                 |                        |                |                    |                |               |
| Must the Record be Present | N                    | (Y/N)            |                 |                        |                |                    |                |               |
| Number                     | Act                  | Link-From        | Print           | End                    | Function       | Link File[.Record] |                |               |
| 1                          | A                    | D1               | D2              |                        | option_present | option             |                |               |
| Header                     | Record Select        | Line Header      | Generate Report | 3* 21                  | System Keys    | Commit Data        | Help           | Previous Menu |

### File Linkages Screen

**To enter the field values:**

| <b>Field</b>                  | <b>Entry</b> | <b>Explanation</b>                                                                                                                                                                                                                                                            |
|-------------------------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Report Name</b>            | prod_opt     | This is the name of the report that you are creating.                                                                                                                                                                                                                         |
| <b>Link Number</b>            | 1            | This screen allows you to define the links one at a time so this number becomes the identifier for the linkage. This identifier is also used on the selection criteria screen if you specify selection criteria for a link.                                                   |
| <b>Action</b>                 | A            | This is the standard action code that allows you to add, change, insert, or delete a linkage specification.                                                                                                                                                                   |
| <b>Link From Line Group</b>   | D1           | A link is always initiated by a particular line group. Any links defined for a line group are initiated before the line is printed. Therefore, the details from the link file record are available for inclusion when the line group is printed.                              |
| <b>Print Line Group</b>       | D2           | A linkage allows you to print another line group after the <i>Link From</i> line group has been printed. This line group is printed for each record selected from the link file.                                                                                              |
| <b>End of Link Line Group</b> | Leave blank. | Once the link is completed and all of the relevant records have been accessed and printed, you can print a further link line group. For example, you could use this line to print some additional summary data. This additional line group is printed at the end of the link. |

**11-20 Developing Greater Reporting Capabilities**



| Field                     | Entry                       | Explanation                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Post-Read Function</b> | <code>option_present</code> | This function is executed each time a record is read from the link file. This allows you to perform a number of actions. For example, you could manually terminate the link by setting the *ENDLINE switch <i>on</i> , or you could manipulate the data retrieved for use in the report. In this report, the function determines if any options have been found for the current product. |
| <b>Link File[.Record]</b> | <code>option</code>         | This is the name of the file that you want to access for the link.                                                                                                                                                                                                                                                                                                                       |
| <b>Index Number</b>       | 2                           | This is the index number that is used to read the link file. A link file can only be read through a key field. The <i>product_no</i> field is defined as key field number 2 for the <i>option</i> file.                                                                                                                                                                                  |

| Field                         | Entry                    | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *KEY=                         | F-<br>product_no.product | This is the key value that is used to access the link file for the first link read. The link ends when the next record read no longer matches the key value, the end of the file is reached, or the switch * ENDLINE is set <i>on</i> .                                                                                                                                                                                                                                                                                                                                                          |
| Must the Record<br>be Present | N                        | This setting takes effect when an unsuccessful attempt is made to read the first link record. When this field is set to <i>N</i> , the file record buffer is cleared and the report continues normally. If this field is set to <i>Y</i> , the buffer is cleared, the line group that initiated the link isn't printed, the link terminates, and the report continues.<br><br>If the file error was not <i>record not found</i> or <i>end of file</i> then the report aborts. Refer to the <i>HP ALLBASE/4GL Developer Reference Manual</i> for further details about specifying critical links. |

**To complete the screen:**

1. Press the **Commit Data** function key to create the file linkage definition.

**Defining the Functions**

You can now define the functions that are called after the link file is read, and before the D2.01 line is printed.

**11-22 Developing Greater Reporting Capabilities**

**To create the required functions:**

1. **Function - *option\_present***

This function is executed each time an attempt is made to link to the *option* file to determine if any option has been found. Create and generate this function now.

```
1 IF F-product_no.product = *P03 THEN EXIT
2 IF *IOSTATUS <> "00000" THEN ON 1;EXIT
3 MOVE F-product_no.product *P03
4 EXIT
```

The *option\_present* function operates as described below.

Line 1            This step determines if this is the first time an option has been read for this product. \*P03 was initialized to spaces in the *start\_of\_report* function to set this condition for the first time the function is executed. If the current product number is equal to the scratch-pad field, this function has already been executed and an option for the product has been read. In this case, the function exits immediately. Otherwise, this function tests if the first option has been found.

Line 2            This step tests the status of \*IOSTATUS. Whenever the link file is read, HP ALLBASE/4GL sets \*IOSTATUS to allow you to determine the success or otherwise of the file access. If it is not zero, user switch number 1 is set *on* and the function exits. The use of this switch will be demonstrated shortly.

Line 3            This step moves the current product number into scratch-pad \*P03 so the test in step 1 can be performed the next time this function is executed.

2. **Function - *print\_option***

This function is executed immediately before the D2.01 line (the option details line) is printed. If no option for the product has been found, it prints the E1 line group instead. Create and generate this function now.

```
1 IF 1 *ON THEN PRINT E1 ; ON *ENDLINE ; OFF 1
```

## 2 EXIT

- Line 1            The *option\_present* function sets the user switch 1 *on* if the file access for the first record fails. This step tests the switch and performs a number of tasks if it is *on*. First the E1 line group is printed. This is a single line that states that no options were found for the product and the product or option files may be corrupted. Then the \*ENDLINE switch is set *on*. This terminates the current line group (D2) so the line is not printed on the report. Then user switch 1 is reset to the *off* state. This is a standard technique to substitute another print line for the current print line.
- Line 3            The function then exits.

You can now create the image of the report.

---

## Painting the Report

Because the report line headers for the different data managers are different, report painting is also different.

If you are developing the KSAM based or the HP TurboIMAGE/iX based application, read on below.

If you are developing the HP ALLBASE/SQL based application, turn to HP ALLBASE/SQL Based Applications.

### **KSAM Based and HP TurboIMAGE/iX Based Applications**

The following illustration shows the layout of the line groups and fields for this report.

|       | 1                                                                                            | 2                                               | 3          | 4          | 5                | 6           | 7          | 8                  | 9                  | 0          |
|-------|----------------------------------------------------------------------------------------------|-------------------------------------------------|------------|------------|------------------|-------------|------------|--------------------|--------------------|------------|
|       | 1234567890                                                                                   | 1234567890                                      | 1234567890 | 1234567890 | 1234567890       | 1234567890  | 1234567890 | 1234567890         | 1234567890         | 1234567890 |
| P1.01 |                                                                                              | Product and Options Stock Report as at AAAAA    |            |            |                  |             |            |                    | Page: NNNN         |            |
| P1.02 |                                                                                              | on the AAAAAAA                                  |            |            |                  |             |            |                    |                    |            |
| D1.01 | Product #                                                                                    | Description                                     |            |            | Supplier         |             | Lead Time  |                    |                    |            |
| D1.02 | AAAAAA                                                                                       | AAAAAAAAAAAAAAAAAAAAAAAAAAAA                    |            |            | NNNNNN           |             | NN         |                    |                    |            |
| D1.03 | Option #                                                                                     | Description                                     |            |            | Cost per Unit    | Qty on Hand |            | Cost of Stock      |                    |            |
| D2.01 | NNN                                                                                          | AAAAAAAAAAAAAAAAAAAAAAAAAAAA                    |            |            | \$NNN,NNN,NNN.NN | NNNNNN AAAA |            | \$NN,NNN,NNN,NN.NN |                    |            |
| T1.01 |                                                                                              | Total cost of Stock for Product AAAAA           |            |            |                  |             |            |                    | \$NN,NNN,NNN,NN.NN |            |
| TF.01 |                                                                                              | Total Cost of Stock for Products AAAAA to AAAAA |            |            |                  |             |            |                    | \$NN,NNN,NNN,NN.NN |            |
| E1.01 | **** No option records found for this product. Product or Option file may be corrupted. **** |                                                 |            |            |                  |             |            |                    |                    |            |

### To paint the line groups and fields:

1. Paint in the literal text items.

You should be able to paint the report with very little assistance.

Some of the data fields do require some explanation. The following paragraphs deal with these, a line group at a time.

#### a. Page Heading Group

The P1.01 and P1.02 lines contain three data fields. The first field at P1.01 column 69 is the report time. Create this as a standard alphanumeric field. When you enter the field details, specify the contents as *\*TIME[1,5]*. This is a substring of the time that only prints the hours and minutes. The field at column 97 is *\*PAGENO*. The field on the next line at column 69 is *\*DATE*.

#### b. Primary Detail Group

The D1.01 to D1.03 lines are printed for each product record found by the report. The D1.01 and D1.03 lines are individual headings for the product details and the option details.

The D1.02 line contains all four fields from the product record. These are *product\_no* at column 2, *description* at column 17, *supplier\_no* at column

59, and *lead\_time* at column 75. Create them using the appropriate field specifications.

c. **Secondary Detail Group**

The D2.01 line is printed for each option record that is found by the linkage to the option file. The first five fields are from the option record. These are *option\_no* at column 4, *description* at column 17, *cost* at column 49, *quantity* at column 68, and *unit\_measure* at column 74. Create these using the appropriate field specifications. Make sure that you specify the option file in the *Field Name/Literal* field for each field specification.

Notice that there is no literal text heading for the *unit\_measure* field. This allows the user to read the quantity and unit of measure as one field.

2. Next, you will modify the cost field image to include numeric editing codes for numeric punctuation and the currency symbol.
3. The numeric editing codes control the format of the field data on the report.
  - a. Position the cursor on the cost field image and press the **Modify Field** function key.
  - b. Enter \$NNN,NNN,NNN.NN.

This overwrites the field specification definition to one that includes the currency symbol and numeric punctuation. It means that data in the cost field image is printed with a leading currency symbol and commas in the correct locations.

4. The last field on the line is the cost of stock. Create this field image manually at column 81, using the **Create Field** function key. The source for the field is a *calculated item* that is totalled into \*TOTALS(1). Enter U-option\_cost in the *Field Name/Literal* field and then 1 in the *Total Number* field.

Although you haven't created this item, you can include it in this report. You will create the calculated item shortly.

## 11-26 Developing Greater Reporting Capabilities

## 5. Subtotal Group

The T1.01 line is printed whenever the value of the first sort field changes. In this case it is the product number field in the product file. The first data field is the product number read from the product file. Create it using the *product\_no* field specification name. The field is created automatically as required.

### Description of the Subtotal Lines

The next few paragraphs provide an outline description of the subtotal lines, subheading lines and the data that they use when they are printed.

A control break occurs when the value of the sort field on the record read from the primary record changes. Therefore, a control break occurs when the next record is read from the file. This may lead you to think that the new details would be used in the subtotal line group. They are not. HP ALLBASE/4GL uses the previous record details for the subtotal line group and then uses the new record details for any subheading line groups. Therefore the subtotal line prints the product number for the product just listed.

The second data field on this line contains the total cost of the options for the product just listed. The field image is identical to the field above it. Enter \*TOTALS(1) in the *Field Name/Literal* field as the name of the data source. If you have used the copy function key to copy the image from the field above it, check to make sure that the other detail fields are satisfactory. In particular, make sure that the automatic totalling field, *Total Number*, is blank.

You may be thinking that using \*TOTALS(1) is correct for the first product listed, but from then on it will give a running total for all options listed. This doesn't occur. HP ALLBASE/4GL keeps a matrix of all 16 \*TOTALS fields. The number of rows in this matrix is always one greater than the number of sort fields you have specified. There is a separate level of \*TOTALS for each control break level, and one for the final value.

In this case \*TOTALS(1) for the level one control break is being accrued for each option listed. When the level one control break occurs, the value of level one control break \*TOTALS(1) is added to \*TOTALS(1) for the final total level. The level one control break \*TOTALS(1) is then reset to zero. The T1 line group refers to the values of the level one control break \*TOTALS fields.

The *HP ALLBASE/4GL Developer Reference Manual* describes the report totalling facilities for reports that use more than one control break.

### **Creating the Final Total Group**

The TF.01 line is printed at the end of the report. It contains the total value of all the stock included in the report. It also lists the range of products that have been included in the report.

#### **To create the TF.01 line:**

1. The first two fields at column 65 and 75 are almost identical. Create their images by using the *product\_no* field specification.
2. Modify their respective field details to change the name of their data sources; the *Field Name/Literal* field should refer to *\*S01* for the field at column 65, and the field at column 75 should refer to *\*S02*.

You can include the screen fields on the current screen in a report.

3. The third field at column 83 is identical to the field above it on line T1.01. Make a direct copy of it to speed up the painting procedure. Remember that this time the value of
4. TOTALS(1) is the final total value accrued over the whole report.

**Extra Line Group** The E1.01 line is only printed by the *print\_option* function when no options have been found for a particular product. It only contains one literal field.

5. Once you have painted each of these lines, exit from the report painter. The report painter automatically saves the image of the report. Before you generate the report, you must define the calculated item used in line D2.01. This is described in the next lesson.

Now turn to the summary of this lesson.

## **11-28 Developing Greater Reporting Capabilities**



## HP ALLBASE/SQL Based Applications

The following illustration shows the layout of the line groups and fields for this report.

```

              1      2      3      4      5      6      7      8      9      10
1234567890123456789012345678901234567890123456789012345678901234567890
P1.01                Product and Options Stock Report as at AAAAA          Page: NNNN
P1.02                on the AAAAAAAA

H1.01 Product #      Description                          Supplier      Lead Time
H1.02 AAAAAA        AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   NNNNNN        NN
H1.03 Option #      Description                          Cost per Unit  Qty on Hand   Cost of Stock
D1.01   NNN         AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA   $NNN,NNN,NNN.NN  NNNNNN AAAA  $NN,NNN,NNN,NNN.NN

T1.01                Total cost of Stock for Product AAAAA  $NN,NNN,NNN,NNN.NN

TF.01                Total Cost of Stock for Products AAAAAA to AAAAAA  $NN,NNN,NNN,NNN.NN

```

### To paint the line groups and fields:

1. Paint in the literal text items now. You should be able to paint the report with very little assistance.

Some of the data fields do require some explanation. The following paragraphs deal with these, a line group at a time.

#### 2. Page Heading Group

The P1.01 and P1.02 lines contain three data fields. The first field at P1.01 column 69 is the report time. Create this as a standard alphanumeric field. When you enter the field details, specify the contents as \*TIME[1,5]. This is a substring of the time that only prints the hours and minutes. The field at column 97 is \*PAGENO. The field on the next line at column 69 is \*DATE.

#### 3. Subheading Group

The H1.01 to H1.03 lines are printed for each product record found by the report. The H1.01 and H1.03 lines are individual headings for the product details and the option details.

The H1.02 line contains all four fields from the product record. These are *product\_no* at column 2, *description* at column 17, *supplier\_no* at column

59, and *lead\_time* at column 75. Create them using the appropriate field specifications.

#### 4. Primary Detail Group

The D1.01 line is printed for each option record that is found by the linkage to the option file. The first five fields are from the *prod\_opt* select list. These are *option\_no* at column 4, *opt\_descript* at column 17, *cost* at column 49, *quantity* at column 68, and *unit\_measure* at column 75. Create these using the appropriate field specifications.

Notice that there is no literal text heading for the *unit\_measure* field. This allows the user to read the quantity and unit of measure as one field.

5. Modify the cost field image. You must modify the cost field image to include numeric editing codes for numeric punctuation and the currency symbol.

The numeric editing codes control the format of the field data on the report.

- a. Position the cursor on the cost field image and press the **Modify Field** function key.
- b. Enter \$NNN,NNN,NNN.NN.

This overwrites the field specification definition to one that includes the currency symbol and numeric punctuation. It means that data in the cost field image is printed with a leading currency symbol and commas in the correct locations.

6. The last field on the line is the cost of stock. Create this field image manually at column 81. The source for the field is a *calculated item* that is totalled into
7. TOTALS(1). Enter U-option\_cost in the *Field Name/Literal* field and then 1 in the *Total Number* field.

Although you haven't created this item, you can include it in this report. You will create the calculated item shortly.

#### 8. Subtotal Group

The T1.01 line is printed whenever the value of the first sort field changes. In this case it is the product number field in the product file. The first data field is the product number read from the product file. Create it using the

### 11-30 Developing Greater Reporting Capabilities

*product\_no* field specification name. The field is created automatically as required.

### **Understanding Subtotal Lines**

The next few paragraphs provide an outline description of the subtotal lines, subheading lines and the data that they use when they are printed.

A control break occurs when the value of the sort field on the record read from the primary record changes. Therefore, a control break occurs when the next record is read from the file. This may lead you to think that the new details would be used in the subtotal line group. They are not. HP ALLBASE/4GL uses the previous record details for the subtotal line group and then uses the new record details for any subheading line groups. Therefore the subtotal line prints the product number for the product just listed.

### **To create the second data field:**

The second data field on this line contains the total cost of the options for the product just listed. The field image is identical to the field above it.

1. Enter \*TOTALS(1) in the *Field Name/Literal* field as the name of the data source.
2. If you have used the copy function key to copy the image from the field above it, check to make sure that the other detail fields are satisfactory. In particular, make sure that the automatic totalling field, *Total Number*, is blank.

You may be thinking that using \*TOTALS(1) is correct for the first product listed, but from then on it will give a running total for all options listed. This doesn't occur. HP ALLBASE/4GL keeps a matrix of all 16 \*TOTALS fields. The number of rows in this matrix is always one greater than the number of sort fields you have specified. There is a separate level of \*TOTALS for each control break level, and one for the final value.

In this case \*TOTALS(1) for the level one control break is being accrued for each option listed. When the level one control break occurs, the value of level one control break \*TOTALS(1) is added to \*TOTALS(1) for the final total level. The level one control break \*TOTALS(1) is then reset to zero. The T1 line group refers to the values of the level one control break \*TOTALS fields.

The *HP ALLBASE/4GL Developer Reference Manual* describes the report totalling facilities for reports that use more than one control break.

### 3. Final Total Group

The TF.01 line is printed at the end of the report. It contains the total value of all the stock included in the report. It also lists the range of products that have been included in the report.

#### To create the final total group:

1. The first two fields at column 65 and 75 are almost identical. Create their images by using the *product\_no* field specification.
2. Modify their respective field details to change the name of their data sources; V-first\_product and V-last\_product.
3. The third field at column 83 is identical to the field above it on line T1.01. Make a direct copy of it to speed up the painting procedure. Remember that this time the value of
4. TOTALS(1) is the final total value accrued over the whole report.
5. Once you have painted each of these lines, exit from the report painter. The report painter automatically saves the image of the report. Before you generate the report, you must define the calculated item used in line D1.01. This is described in the next lesson.

---

## Summary

In this lesson you created a report that accessed two different data files. In creating this report, you used the following report facilities.

- A start of report function.
- Sort field definitions.
- Record selection criteria (HP TurboIMAGE/iX and KSAM only).
- File linkages and a post link read function (HP TurboIMAGE/iX and KSAM only).
- A before print function associated with a report line.

### 11-32 Developing Greater Reporting Capabilities

### **Start of Report Functions**

A start of report function allows you to display a screen to obtain values for reporting from the user. The start of report function is the only report function that can display a screen.

### **Sort Field Definitions**

The sort field definitions for a report define the report control breaks. A control break occurs whenever the value in a sort field changes from one record to the next.

At a control break, HP ALLBASE/4GL prints subtotal and subheading line groups. These line groups can contain subtotal values that have been accumulated in one of the \*TOTALS(n) communication area fields.

You can define sorting fields even if the primary report file is in the correct order. HP ALLBASE/4GL still uses the sort fields to define the control breaks for the report.

### **Record Selection Criteria**

Report record selection criteria allow you to select records that are printed on the report. The selection criteria specify a range of values for a field on the record read from the report primary file or a link file.

If the value in the specified field does not fall within the specified range, the record is not included in the report.

If you are creating the HP ALLBASE/SQL based application, you can also select records using a SELECT command in an SQL logic block.

### **File Linkages**

If you are creating the HP ALLBASE/SQL based application, you can perform file linkages by creating an HP ALLBASE/SQL select list.

For other data managers, you can perform file linkages, using the File Linkages screen. File linkages allow you to retrieve data from a file other than the primary report file. File linkages are defined on a line group basis.

Before HP ALLBASE/4GL prints the line group, it performs the initial read of the link files defined for the line group. The data retrieved by the link read can be included in the initiating line group. The data can also be included in link print line groups.

You can specify that a function is executed after the link file read.

### **Print Line Functions**

HP ALLBASE/4GL allows you to execute a function before and after each physical report line is printed. In this report, you used a before print function to substitute a different print line if an option file read fails to find any options for a product.

### **Calculated Items**

This lesson also introduced calculated items. You will complete the definition of the *option\_cost* calculated item in the next lesson.

---

## Lesson 28-Defining Calculated Items

---

### Objectives

When you have completed this lesson you will have learned how to define and generate a calculated item.

This calculated item completes the requirements of the report you defined in the previous lesson.

---

### Calculated Items

A calculated item is a special type of variable. A calculated item implicitly refers to a function. When HP ALLBASE/4GL encounters a reference to a calculated item, it automatically executes the function to determine the value of the calculated item.

A calculated item can be defined as a single CALC command, or it may be extended to use as many lines of logic in the function as you require.

#### Menu Path

To access the calculated items screen:

1. From the main menu, select the *Dictionary* option.
2. Select the *Storage Items* option.
3. Choose *Calculated Items*.

## Screen Description

This screen allows you to define a calculated item. You will define the item's basic characteristics and a single CALC statement to derive the value of the calculated item. Defining a calculated item automatically creates a function of the same name. This function is automatically executed each time the item is accessed.

If the item you are defining requires more than a single CALC statement, you can subsequently call up the function in the function details screen. A calculated item defined this way is known as an extended calculated item.

The screen below shows the completed screen after defining a calculated item for the KSAM version of the application.

```
Developer          Calculated Items          calculated_items

Item Name          option_cost
Length             12
Edit Code          N (N/A/U/K/N/S/Q/D/T)
Number of Decimal Places 2
Pad Character      .

CALC F-cost.option x F-quantity.option = U-option_cost

Description
option_cost
AUTHOR: developr
This calculation determines the cost of stock for one option
Last modification:   Date 4/27/92      Time 10:47:01

Variable Function Numeric AlphaNum 8* 36 System Commit Help Previous
Details Constant Constant Keys Data Menu
```

**Calculated Items Screen**

## 11-36 Developing Greater Reporting Capabilities



**To enter the field values:**

| <b>Field</b>                    | <b>Entry</b> | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Item Name</b>                | option_cost  | This is the name of the calculated item that you are defining. HP ALLBASE/4GL automatically creates a function with the same name for the necessary logic that is needed behind the item. Even if the calculated item only requires one line of logic to calculate its value, HP ALLBASE/4GL still creates the function. A calculated item cannot have the same name as an existing function in the same application. |
| <b>Length</b>                   | 12           | The length of the calculated item can be up to 99 characters.                                                                                                                                                                                                                                                                                                                                                         |
| <b>Edit Code</b>                | N            | This is a standard edit code.                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Number of Decimal Places</b> | 2            | This specifies the maximum number of decimal places assigned to the item. It is only relevant for numeric fields.                                                                                                                                                                                                                                                                                                     |
| <b>Pad Character</b>            | Leave blank. | As for variables, alphanumeric calculated items are left justified and numeric items are right justified. The pad character entered here replaces any leading or trailing spaces remaining as a result of the justification.                                                                                                                                                                                          |

| Field                   | Entry                                                                                                                                                                                                                                                    | Explanation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>The CALC command</b> | <p>For KSAM based and HP TurboIMAGE/iX based applications, enter:</p> <pre>F-cost.option x F-quantity.option = U-option_cost</pre> <p>For HP ALLBASE/SQL based applications, enter:</p> <pre>F-cost.prod_opt x F-quantity.prod_opt = U-option_cost</pre> | <p>When you leave the pad character field you will see that the word <i>CALC</i> is displayed to the left of the input area occupied by the cursor. This input area allows you to enter the statement to calculate the value of the item. Refer to the <i>HP ALLBASE/4GL Developer Reference Manual</i> for further details about the <i>CALC</i> command.</p> <p>Note that the destination of the <i>CALC</i> command is the calculated item itself. This is how a value is assigned to a calculated item.</p> <p><b>Note</b> If you extend a calculated item, you can use any logic block commands including a visit to another function. However, the only function that can assign a value to a calculated item is the function with the same name as the calculated item. If you attempt to assign a value to a calculated item from any other source, the application will abort at run-time.</p> <p>If you don't want to use a <i>CALC</i> statement, or wish to extend this function you can leave the <i>CALC</i> statement blank. You can then define the logic for the item by calling up the function with the function details screen.</p> |

**Description** Enter a suitable description.

**To complete the entries:**

1. Press the **Commit Data** function key when you have finished creating the calculated item.

**11-38 Developing Greater Reporting Capabilities**

A calculated item automatically creates a function of the same name. Pressing the **Commit Data** function key generates the function automatically. If any errors occur during the generation operation, they are displayed in the normal way. Correct any errors using either the calculated item screen or the function details screen.

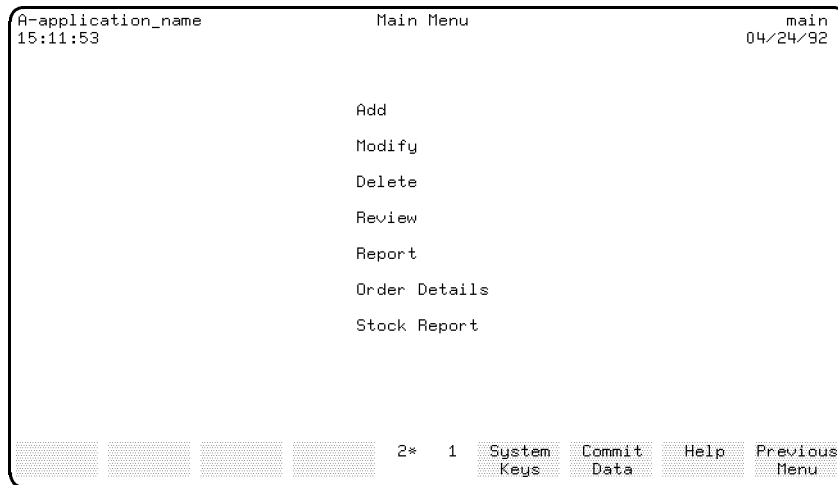
You may wish to examine the logic block created by the CALC item by looking at it using the function details screen.

## Generating the Report

Once you have defined the calculated item, generate the *prod\_opt* report. Now you will need to alter the main menu so that you can execute the report.

## Altering the Main Menu

This is what the main menu will look like when you have repainted it.



### Repainted Main Menu

#### To alter the main menu:

1. Call up the *main* menu in the screen painter.
2. Add the following action item to the screen.

Where you place it on the screen is up to you. The screen image above is one suggested format for the menu.

| Type       | Name     | Text         |
|------------|----------|--------------|
| R (report) | prod_opt | Stock Report |

3. When you have defined this item, exit from the painter.

HP ALLBASE/4GL saves the menu as you exit.

### Running the Report

When you have altered the main menu, you may run your application and test the report. You defined that the report printer destination was printer 1. Ask your HP ALLBASE/4GL administrator what has been defined as printer 1 for your system, or, if you have access to the *administ* application, you may check for yourself by selecting System Specifications on the main menu, and then System Definition on the System-Wide Specifications menu. This displays the System Definition screen, which lists the printers defined for HP ALLBASE/4GL.

The *training* application is now functionally complete.

---

## Summary

In this lesson you created a calculated item. Calculated items have the following properties.

- A calculated item is similar to a variable, except that its value is determined by executing a function every time the calculated item is referenced.
- Calculated items can be simple calculated items, or extended calculated items. In the simple form, the value of the item is determined by a single CALC command. In the extended form, the value of the calculated item is determined by a function containing more than one command, or containing a command other than a CALC command.

### 11-40 Developing Greater Reporting Capabilities

- Creating a calculated item automatically creates a function with the same name. HP ALLBASE/4GL generates the function automatically when you commit the calculated item.
- The value of a calculated item can only be assigned by the function with the same name as the calculated item. If you attempt to move a value into a calculated item from any other source, the application will abort.

---

## Where You Are Now

Congratulations! You have completed the HP ALLBASE/4GL Developer Self-Paced Training Guide. (Actually, there are still the Self Test Questions for this chapter on the following pages.)

While you have been developing the *training* application in this guide you have seen how to define the majority of items in HP ALLBASE/4GL. You have also seen how HP ALLBASE/4GL runs applications, and how HP ALLBASE/4GL can do a lot of work for you during the application development phase.

Obviously there are many more ways to use HP ALLBASE/4GL than those outlined in this training guide. You have probably started referring to the *HP ALLBASE/4GL Developer Reference Manual*. By now you should be familiar with its layout and content. We suggest that you now start experimenting and solving real life problems that are specific to your data processing needs.

---

## Self Test Questions

**Question 1.** Can scratch-pad fields accept alphanumeric values?

**Question 2.** Is the following statement true or false?

For every generated calculated item, a function of the same name exists.

**Question 3.** What is the relationship between sort fields and report control breaks.

**Question 4.** What does the report sorting screen allow you to specify?

**Question 5.** When would you use a D2 line group in a report?

---

## Answers

**Answer 1.** Yes, scratch-pad fields can accept both numeric and alphanumeric values.

**Answer 2.** True. Whenever you generate a calculated item, HP ALLBASE/4GL creates and generates a function of the same name.

**Answer 3.** Each time the value in a sort field changes, from one record to the next, a control break for the sort level occurs.

**Answer 4.** The report sorting screen allows you to specify the following items.

- Up to eight sorting levels.
- Whether records are to be sorted in ascending or descending order.
- Up to eight control break levels.

**Answer 5.** A D2 line group would be used to print the details read from a link file. The report generator prints link detail line groups when they are specified on the file linkages screen.

FINAL TRIM SIZE : 7.0 in x 8.5 in



# A

## Starting a New Application

---

This appendix describes the procedures for creating an application definition in the HP ALLBASE/4GL *administ* application. A developer cannot develop an application until it has been defined in the *administ* application.

This chapter is written for the system administrator. If you do not have access to the administrator application, you cannot complete the application definition.

---

### Before You Start

This training guide allows the user to create an application using one of three data managers. The user may use one of the following data managers:

- KSAM data files.
- An HP ALLBASE/SQL database.
- An HP TurboIMAGE/iX database.

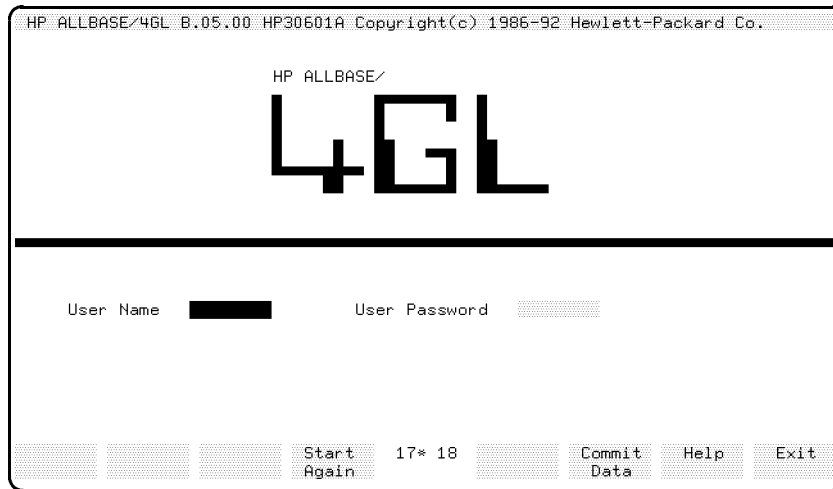
Before the application can be defined, you must determine which data manager will be used.

You can only use an HP ALLBASE/SQL database or an HP TurboIMAGE/iX database if you have the data manager on your system. The KSAM data manager is provided with HP ALLBASE/4GL.

---

## The Sign-On Screen

The sign-on screen is the first screen users see after starting up HP ALLBASE/4GL.



### Administrator Sign-On

Enter the administrator user name `administ`, and then press `Return`.

HP ALLBASE/4GL checks that you have entered the correct administrator user name. If a password is required, you will be prompted to enter it. The password does not appear on the screen as you enter it.

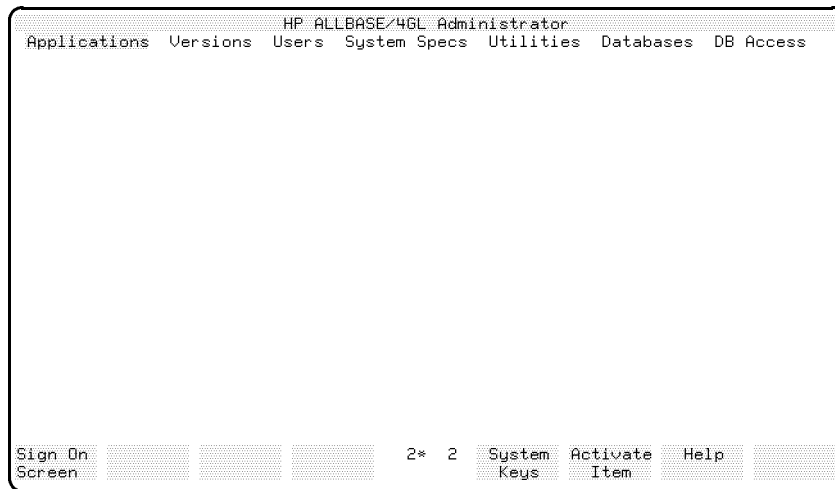
Press the `Commit Data` function key to run the administrator application.

## A-2 Starting a New Application

---

## Administrator Main Menu

When the administrator application starts, HP ALLBASE/4GL displays the administrator main menu.



## Developer Validation

Before users can use the developer to develop an application, they must be registered in the HP ALLBASE/4GL administrator as a developer user. The screens used to define HP ALLBASE/4GL users are accessed from the user validation submenu of the administrator main menu. To display the developer validation screen, select the *User Definitions and Security* option from the main menu. This displays the user validation menu. Select the *Developer Validation* option on this menu to display the developer validation screen.

You may also want to assign the user an end user name. This allows the user to run the application as an end user, rather than as a developer. To display the *End User Validation* screen, select the *End User Validation* option from the user validation menu. Refer to the *HP ALLBASE/4GL Developer Administration Manual* if you need more information about this screen.

| Administrator       |                                                                                                     | Developer Validation |        | developer |               |
|---------------------|-----------------------------------------------------------------------------------------------------|----------------------|--------|-----------|---------------|
| Developer Name      | username                                                                                            |                      |        |           |               |
| Current Password    |                                                                                                     |                      |        |           |               |
| New Password        |                                                                                                     |                      |        |           |               |
| Repeat New Password |                                                                                                     |                      |        |           |               |
| Description         | username<br>Developer name for new developer using ALLBASE/4GL to develop the training application. |                      |        |           |               |
| Last Modification:  | Date                                                                                                | Time                 |        |           |               |
| End User            | Menu                                                                                                | 7* 31                | System | Commit    | Help          |
| Validatn            | Security                                                                                            |                      | Keys   | Data      | Previous Menu |

### Completing the Screen

Before you complete this screen, check the details. Your site may have established standards for developer names, user names, or passwords.

#### Developer Name.

Enter the selected developer name in this field. Developer names can use alphabetic characters, 0 to 9, and \_ (underscore). The name must start with an alphabetic character. HP ALLBASE/4GL is case sensitive with respect to developer names. The names *administ* and *developr* are reserved names, and cannot be used as developer names.

**Current Password.** No entry is possible at this time as this is a new entry. This field is only active if you have already defined a password for this developer name.

**New Password.** This is optional.

If you want to assign a password to this developer name, enter the password in this field. Passwords can be up to eight characters long, and can use alphabetic characters, 0 to 9, and \_ (underscore). The password must start with an alphabetic character. HP ALLBASE/4GL is case sensitive with respect to passwords.

### A-4 Starting a New Application

**Repeat New Password.** This is optional.

If you entered a password in the *New Password* field above, you must repeat exactly the same password in this field. If you don't re-enter exactly the same password, HP ALLBASE/4GL generates an error message, and you must re-enter the password in the *New Password* field.

**Description.** Enter a brief description of the developer used for documentation purposes.

Press the **Commit Data** function key to create this developer definition. You can now go on to complete the application definition screen.

---

## Training Application Definition

The following paragraphs describe the procedure for creating the application definition for the *training* application. Developers may choose to develop their application using one of three data managers; KSAM data files, HP ALLBASE/SQL databases, or HP TurboIMAGE/iX databases. Make sure that you complete the correct instructions.

The application definition screen allows you to define the essential details about the administration of an application. You must define the application name, password, and the users or user groups. You must also define the name and the type of the first action to be executed when the application starts.

You don't have to strictly adhere to the following suggested entries. They are the minimum requirements. There may be some administrative standards implemented at your site that require you to alter some of the values specified here. The following text draws attention to fields that must **not** be changed.

```

Administrator          Application Definition          applic_defn
Application            training
Current Password
New Password
Repeat New Password
Current Development Security Code
New Development Security Code
Repeat New Security Code
Initial Action Name   main          Type   M (M/P)
SQL Owner Group
SQL Database Name

Valid Users/Groups
*ALL

Description
training application
PURPOSE: This is the training application for new developers
who are completing the ALLBASE/4GL self-paced training
course.
Last Modification:   Date   3/12/92   Time   12:44:20

Version  Utility  User  System  6* 21  System  Commit  Help  Previous
Defn.   Menu   Menu  Menu   Keys  Data   Menu   Menu

```

## Entering the Field Values

**Application.** Enter `training`

This is the name that developers and end users will use to access this application.

You don't need to use the name *training*. However, if you do use a different name, ensure that you use that name in place of any reference to the application name *training* throughout the *Developer Self-Paced Training Guide*. Remember that the names *administ* and *developr* are reserved names, and cannot be used as application names.

## Application Password Definition

The next three fields allow you to define a user password for this application. If you do define a password, any end user wanting to use it must enter the password before gaining access to the application.

**Current Password.** No entry is possible in this field as no password exists for a new application.

## A-6 Starting a New Application

This field is bypassed as no password is set when the application is first defined. When the application does have a password you must enter it in this field before you can change the existing password.

**New Password.** Leave this field blank.

If you want to enter a password for this application then you can enter a password in this field. Passwords can use alphabetic characters, 0 to 9, and \_ (underscore). The first character of a password must be alphabetic. As with all other names, HP ALLBASE/4GL is case sensitive with respect to passwords.

**Repeat New Password.** Leave this field blank.

If you want to enter a password for this application then you should re-enter the password in this field. If the two password entries don't match, you must re-enter the new password in the *New Password* field.

### **Development Security Code**

The next three fields allow you to define a development security code for the application. If you do define a security code, any developer wishing to change secured components in the application must enter the correct security code when signing on to the application.

**Current Development Security Code.** No entry is possible in this field.

This field is only active if a security code already exists for the application.

**New Development Security Code.** Leave this field blank.

This is where you can enter a security code for this application. If you do enter a code, it is not echoed to the screen.

**Repeat New Security Code.** Leave this field blank.

If you enter a new security code, you must repeat it in this field. If the two entries don't match, you must re-enter the *New Development Security Code*.

## Other Fields

**Initial Action Name.** Enter `main`

This field defines the name of the first item in the application that HP ALLBASE/4GL executes when the application starts. You must enter `main` for this application.

**Type.** Enter `M`

This field specifies the action type for the initial action. It can be either a **menu** or a **process**.

**SQL Owner Group.**

If the developer is going to develop the KSAM based or HP TurboIMAGE/iX based application, leave this field and the next field blank.

If the developer is going to develop the HP ALLBASE/SQL based application, enter `sqlgrp`

HP ALLBASE/4GL allows application developers to create base tables in the application database environment. HP ALLBASE/4GL also creates a database module when an application is generated. HP ALLBASE/SQL requires that base tables and modules have an owner.

HP ALLBASE/4GL transfers ownership of all base tables and the module for this application to the owner group specified in this field.

You can enter a different name in this field if required. However, if you do use a different owner group name, all references to the owner group *sqlgrp* in this manual must be replaced with the correct owner group name.

---

### Note



You cannot change the SQL owner group name for an application after development of the application has commenced.

---

**SQL Data Base Name.**

If the developer is going to develop the HP ALLBASE/SQL based application, enter `sqldbase`

Otherwise, leave this field blank.

## A-8 Starting a New Application



This is the name of the database environment for the application. The name of a database environment is the name of the DBEConfile for the database.

HP ALLBASE/4GL uses a MPE/iX variable HP4SQLPATH to identify the account and group containing the database DBEConfile. Refer to the *HP ALLBASE/4GL Developer Administration Manual* for more information about the MPE/iX variables used by HP ALLBASE/4GL.

Once again, you can use a different name for the database environment provided you substitute the new name for all references to *sqlbase* in this guide. You cannot change the database environment name after application development has commenced.

### Users and Groups List

The next 12 fields list the valid user names or groups who can access a particular application. Only the developers and end users listed here can access this application.

**Valid Users/Groups.** Enter \*ALL.

\*ALL is a special entry specifying that anyone gaining access to HP ALLBASE/4GL may access this application. You may wish to enter specific user names in these fields.

Leave the remaining Valid Users/Groups fields blank.

**Description.** Enter a suitable description of the application:

```
Training application
This is the training application for ...
```

This is a brief description of the application that is used for documentation purposes.

### When the Entries are Complete

Press the **Commit Data** function key to complete this screen.

If you are defining the application for KSAM developers, you have completed the application definition, and users can proceed with developing the application. Return to the main menu by pressing the **Previous Menu** function key, and then press the **Sign on Screen** function key to return to the sign-on screen.

If you are defining the application for HP ALLBASE/SQL developers, continue reading below.

If you are defining the application for HP TurboIMAGE/iX developers, turn to HP TurboIMAGE/iX Database Creation.

---

## SQL Application Definition

The following paragraphs describe the procedure to define the HP ALLBASE/SQL database for the *training* application.

### SQL Database Definition

HP ALLBASE/4GL does not create database environments directly. You must create the database environment using HP ISQL before application development can commence.

The database environment for an application must contain at least one DBEfileset and one DBEfile.

The commands listed below show the procedure for creating a database environment that is suitable for the *sqltrain* application. This database is the smallest workable size for developing this application. You may want to enlarge or modify the database if you have sufficient free disk space.

To create the database environment, you must start the HP ISQL program in the account and group for the database environment DBEConfile. Typically, the command to start HP ISQL is simply ISQL.

In the dialog below, user input is shown underlined for clarity. The characters `isql=>` represent the normal HP ISQL prompt, and the character `>` is the HP ISQL continuation prompt that appears when a command extends beyond one line.

Enter the following commands at the HP ISQL prompt. Note that you must terminate each command with a semicolon (;).

```
isql=> START DBE 'sqldbse' MULTI NEW  
> TRANSACTION = 6,
```

### A-10 Starting a New Application

```

> DBEFILEO DBEFILE sqlDBEO
> WITH PAGES = 200, NAME = 'sqlFO',
> LOG DBEFILE sqlDBElog1
> WITH PAGES = 288,
> NAME = 'sqllog1';

isql=> CREATE DBEFILESET PRODFS;
isql=> CREATE DBEFILESET ORDRFS;
isql=> CREATE DBEFILESET OPTFS;

isql=> CREATE DBEFILE prodDBEfile WITH PAGES = 50,
> NAME = 'prodfile', TYPE = MIXED;

isql=> CREATE DBEFILE ordrDBEfile WITH PAGES = 30,
> NAME = 'ordrfile', TYPE = MIXED;

isql=> CREATE DBEFILE optDBEfile WITH PAGES = 20,
> NAME = 'optfile', TYPE = MIXED;

isql=> ADD DBEFILE prodDBEfile TO DBEFILESET PRODFS;
isql=> ADD DBEFILE ordrDBEfile TO DBEFILESET ORDRFS;
isql=> ADD DBEFILE optDBEfile TO DBEFILESET OPTFS;
isql=> commit work;
isql=> exit;

```

### Database Access

The MPE/iX user who creates a database environment automatically becomes the database administrator. This user must grant appropriate access authority to other users.

Application developers must have CONNECT authority and RESOURCE authority for the application database environment. Application developers must also be members of the SQL owner group for the application.

---

## HP TurboIMAGE/iX Database Creation

The following paragraphs describe the procedure to define the HP TurboIMAGE/iX database for the *training* application.

The first step is to define the database. Run HP TurboIMAGE DBSCHEMA using the following database definition.

```
BEGIN DATA BASE TRAIN, LANGUAGE:AMERICAN;  
PASSWORDS: 10 DEVELOP;
```

```
ITEMS:
```

```
SUPPLIER-NO,      I1  (/10);  
PRODUCT-NO,      X6  (/10);  
DESCRIPTION,     X30 (/10);  
LEAD-TIME,       I1  (/10);  
ORDER-NO,        I1  (/10);  
OFFICER-NAME,   X30 (/10);  
QUANTITY,        I1  (/10);  
UNIT-MEASURE,   X4  (/10);  
COST,            P12 (/10);  
ORDER-DATE,     X8  (/10);  
DELIVERY-DATE, X8  (/10);  
OPTION-KEY,      X10 (/10);  
OPTION-NO,       X4  (/10);
```

```
SETS:
```

```
NAME:      PRODUCT, MANUAL (/10);  
ENTRY:    PRODUCT-NO (2),  
          DESCRIPTION,  
          SUPPLIER-NO,  
          LEAD-TIME;  
CAPACITY: 211;
```

```
NAME:      OPTION-KEY, AUTOMATIC (/10);  
ENTRY:    OPTION-KEY (1);  
CAPACITY: 211;
```

### A-12 Starting a New Application

NAME: OFFICERS, AUTOMATIC (/10);  
ENTRY: OFFICER-NAME (1);  
CAPACITY: 211;

NAME: ORDER-NO, AUTOMATIC (/10);  
ENTRY: ORDER-NO (1);  
CAPACITY: 211;

NAME: ODATE, AUTOMATIC (/10);  
ENTRY: ORDER-DATE (1);  
CAPACITY: 211;

NAME: DDATE, AUTOMATIC (/10);  
ENTRY: DELIVERY-DATE (1);  
CAPACITY: 211;

NAME: ORDER, DETAIL (/10);  
ENTRY: ORDER-NO (ORDER-NO),  
OFFICER-NAME (OFFICERS),  
PRODUCT-NO (!PRODUCT),  
QUANTITY,  
UNIT-MEASURE,  
COST,  
ORDER-DATE (ODATE),  
DELIVERY-DATE (DDATE);  
CAPACITY: 200;

NAME: OPTION, DETAIL (/10);  
ENTRY: OPTION-KEY (OPTION-KEY),  
PRODUCT-NO (!PRODUCT),  
OPTION-NO,  
DESCRIPTION,  
COST,  
QUANTITY,  
UNIT-MEASURE; CAPACITY: 200;

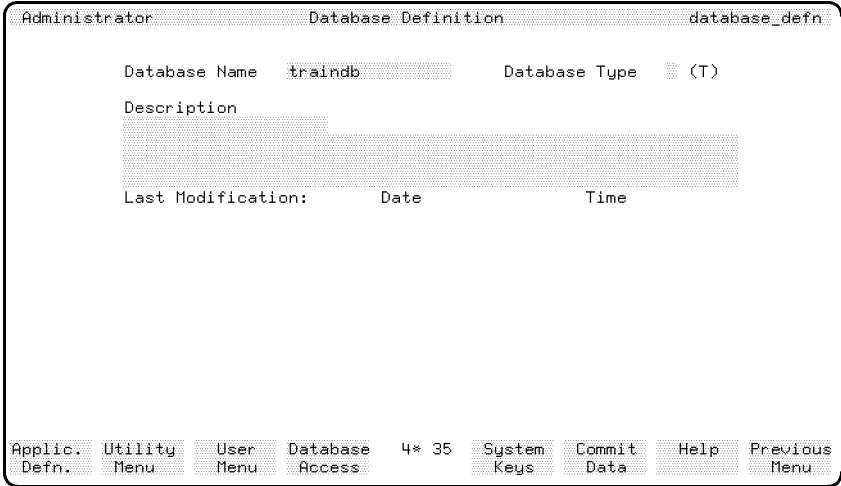
END.

Then create the database, using HP TurboIMAGE DBUTIL. For more information about creating the database, refer to the *HP TurboIMAGE/iX Database Management System Reference Manual*.

Once you have created the database, you must define it within the HP ALLBASE/4GL administrator application.

### Defining the Database

The following paragraphs describe the procedure for defining the database within HP ALLBASE/4GL. This screen allows you to specify the name, group, and account of the HP TurboIMAGE/iX database and allocate an HP ALLBASE/4GL name for the database.



**Database Definition Screen**

**To enter the field values:**

Complete the database definition screen using the following information.

### A-14 Starting a New Application

| Field         | Entry                                              | Explanation                                                                                                                                                                                      |
|---------------|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Database Name | traindb                                            | This is the name of the database as it will be known within HP ALLBASE/4GL.                                                                                                                      |
| Database Type | Accept the default value of T.                     | This field indicates the type of database that you are defining. Currently, you can only define HP TurboIMAGE/iX databases. When you commit this field, an HP TurboIMAGE/iX window is displayed. |
| Description   | Enter a description of the database in this field. |                                                                                                                                                                                                  |
| External Name | Enter TRAIN                                        |                                                                                                                                                                                                  |

The screenshot shows a terminal window titled "HP TurboIMAGE/iX Window". At the top, the label "External Name" is followed by a text input field containing the value "TRAIN". Below the input field is a menu bar with the following items: "Applic. Defn.", "Utility Menu", "User Menu", "Database Access", "7\* 12", "System Keys", "Commit Data", "Help", and "Previous Menu".

### HP TurboIMAGE/iX Window

If the database is not in the account and group identified by the HP4TIPATH system variable, you should enter the fully qualified name of the database root file in this field (that is, including the group and account in the name).

#### To complete the entries:

1. Press the **Commit Data** function key to complete this screen.

Your next step is to give the training application access to this database.

## The Parameters for Database Access Screen

Go to the parameters for database access screen to give the training application access to the traindb database.

```

Administrator      Parameters for Database Access      database_access

Application training
Accessible Databases  Type  Parameters

Database Name      Action (A/C/D)
traindb

Applic. Database      18* 12  System  Commit  Help  Previous
Defn.  Defn.
  
```

### Parameters for Database Access

To enter the field values:

| Field         | Entry                                 | Explanation                                                                                                    |
|---------------|---------------------------------------|----------------------------------------------------------------------------------------------------------------|
| Application   | training                              | This is the name of the application to which you wish to allow database access.                                |
| Database Name | traindb                               | This is the name of the HP TurboIMAGE/iX database. When you commit this field, a password window is displayed. |
| Action        | Accept the default action of Add (A). |                                                                                                                |
| Password      | DEVELOP                               |                                                                                                                |

## A-16 Starting a New Application



|               |                |        |                |               |
|---------------|----------------|--------|----------------|---------------|
| Database Name | traindb        | T      | Action (A/C/D) | A             |
|               |                |        | Password       | DEVELOP       |
| Applic. Defn. | Database Defn. | 20* 62 | System Keys    | Commit Data   |
|               |                |        | Help           | Previous Menu |

### Password Window

This HP TurboIMAGE/iX database password supplies all developers and users of the training application with write access to all parts of the database.

#### To complete the entries:

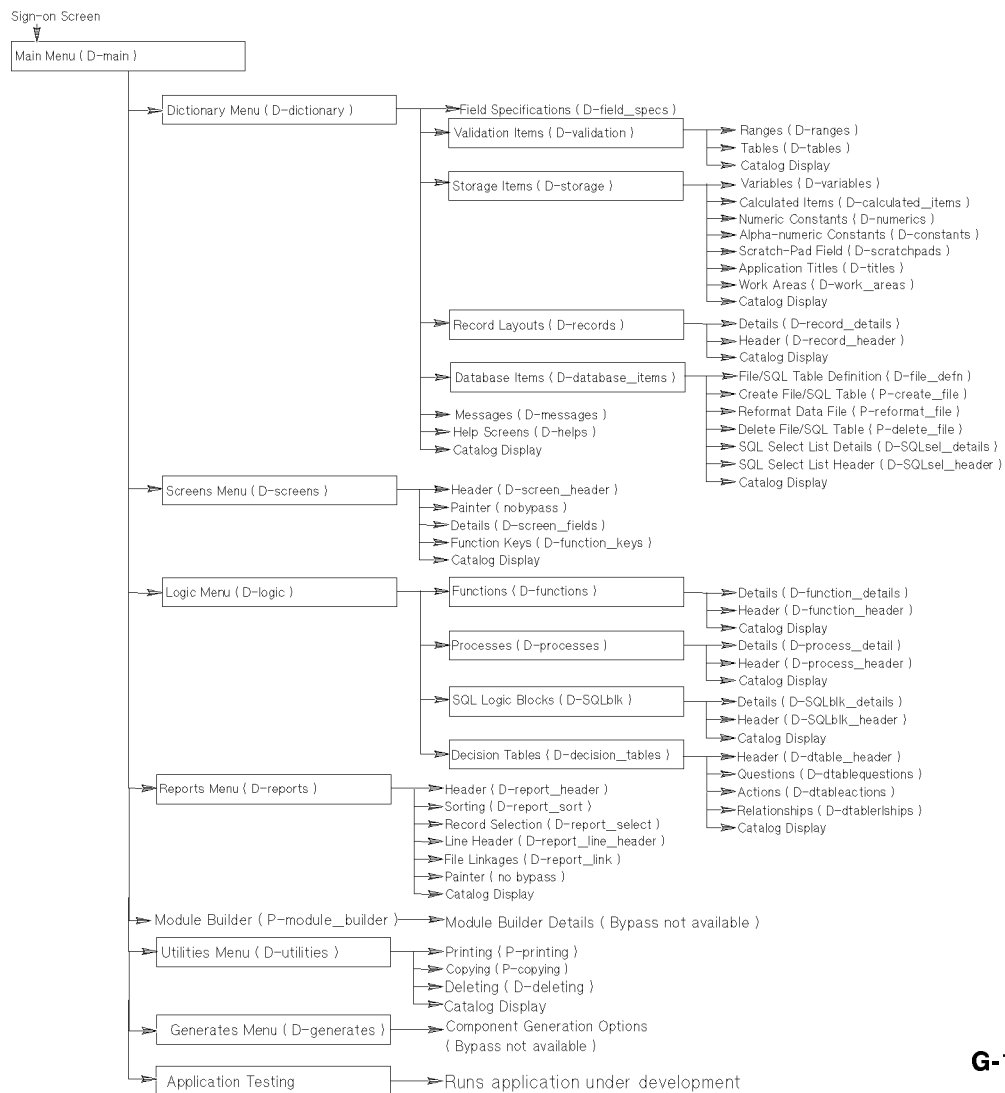
1. Press the **Commit Data** function key to complete this screen.

You have now completed the application definition and database definition required for developers to begin developing the training application.

FINAL TRIM SIZE : 7.0 in x 8.5 in



# Developer Reference Chart



FINAL TRIM SIZE : 7.0 in x 8.5 in

## Glossary

---

This glossary explains the meaning of some terms and words used within this manual and in HP ALLBASE/4GL. In some cases, the terms used in HP ALLBASE/4GL may differ in meaning from the same terms used in a conventional programming environment.

### A

**Abort.** A message type that informs the user that a serious error has been detected. The HP ALLBASE/4GL session terminates after the user acknowledges the message.

**\*ABSENT.** VALIDATE command argument that is used to check if the specified KSAM file record field does not exist.

**Administrator.** The person who controls the HP ALLBASE/4GL site configuration and controls access by developers and users.

**administ.** The name of the administrator application. It is a reserved name.

**administrator.** The application used by the system administrator to control the system-wide site configuration. Note that this name has a lowercase "a".

**\*ALL.** OFF and ON logic command argument for referencing all user switches 1 through 8.

**\*ALPHA.** IF and IFLOOP logic command argument used for checking if a field contains only alphabetic characters.

### B

**BACKGRND.** Logic command for executing a process in background mode. Only an HP ALLBASE/4GL process can be executed in this manner. The process may execute any facility except displaying a screen. A background

process cannot send messages to its originating terminal or accept input from the terminal.

**\*BLANK.** IF and IFLOOP command argument used to check if a field contains only space characters.

**\*BUFFER.** FILE command argument used to clear a file record buffer. This command also unlocks a file record that has been locked to the file buffer.

**\*BYPASS.** A switch used by the report generator. When the switch is set *on*, it indicates that a record is not to be processed.

## C

**CALC.** Logic command for performing a mathematical operation.

**Calculated Item.** A variable that is evaluated each time it is referenced. In the simple form, the calculated item is the result obtained by evaluating a CALC logic command. In the extended form, HP ALLBASE/4GL executes a function each time the calculated item is referenced. In this case, the calculated item function assigns a value to the calculated item.

**CHECK.** Logic command for matching data against an indexed file record or table. The CHECK command places the matching position number into the communication area field \*PASS.

**CLEAR.** Logic command for clearing all or part of the screen or scratch-pad.

**\*CLOSE.** FILE command argument used to close the nominated file.

**\*COMMIT.** Command argument used with the EXIT command in an after function on a data screen input field, a prior function on a display-only field on a data screen, or a function called from a function key on a data screen. This argument terminates the current screen in the same way as the I-COMMIT internal routine.

**Communication Area Field.** A field addressable by HP ALLBASE/4GL and the logic of the application. Communication area fields may be either alterable or read only. Any non-numeric communication area field may be referenced using the substring conventions. Numeric communication area fields are held as numbers and may not be referenced by substring.

## Glossary-2

**\*COUNT(n).** Alterable communication area fields used primarily in the report generator for counting the number of times a nominated report line is printed. HP ALLBASE/4GL has five line counter fields, identified as \*COUNT(1) through \*COUNT(5). These fields are treated as numeric.

**\*CROSS(n).** Alterable communication area fields used primarily in the report generator for across-the-page totalling of numeric fields. HP ALLBASE/4GL has five cross add fields, \*CROSS(1) through \*CROSS(5). These fields are treated as numeric.

## **D**

**Data screen.** The normal HP ALLBASE/4GL screen on which the user can enter data, and on which the system displays information. Data screens can contain a scrolling area or a window or both. The SCREEN logic command or the D- action on a menu initiates display of a data screen.

**Data item.** The smallest accessible data element in an HP TurboIMAGE/iX database. Within HP ALLBASE/4GL, data items are defined as field specifications.

**Data entry.** An ordered set of related data items within an HP TurboIMAGE/iX database. Within HP ALLBASE/4GL, data entries are defined as record layouts.

**Data set.** A collection of data entries, or records, within an HP TurboIMAGE/iX database.

**\*DATE.** Read-only communication area field containing the current date. This field may be referenced by substring.

**Date Format.** Each HP ALLBASE/4GL site has one date format (specified by the system administrator) that applies to all developers and users. The format is either the US date format MM/DD/YY, or the European date format DD/MM/YY. Regardless of the date format, HP ALLBASE/4GL stores all dates on file records in the format YYMMDD. The chosen system-wide data format controls the presentation of dates on screens and reports.

**DECISION.** Logic command for executing a decision table.

**Decision table.** A decision table consists of up to 8 questions, up to 31 different combinations of outcomes for the questions, and up to eight

actions. For each combination of outcomes, the decision table specifies which actions are to be performed, and the sequence in which they are performed.

**DEFINE.** Logic command that allows you to define a macro in a logic block. The DEFINE command substitutes an expression for a one to four character string surrounded by \% signs.

**\*DELETE.** FILE command argument that specifies file record deletion.

**developr.** The name of the developer application.

**DISPLAY.** Logic command for clearing and displaying data within the scroll area of a data screen.

**DM.** A prefix to all logic commands that are specific to a data manager. Currently, HP ALLBASE/4GL offers four DM commands: DM IMAGE \*CLOSE, DM IMAGE \*LOCK, DM IMAGE \*MODE, and DM IMAGE \*UNLOCK.

## **E**

**Edit code.** A code assigned to a field specification, a screen field, or a report field that defines the type of data stored or displayed in the field.

**\*ENDFIELD.** A switch used by the screen processing logic. A function called from a function key can set this switch *on*. If \*ENDFIELD is *on*, normal input processing for the current field is terminated, and processing continues at the next field on the screen when the function exits.

**\*ENDLINE.** A switch used by the report generator. When this switch is *on*, it indicates that no more components of the current line group are to be printed, or no more records are to be read from the current link file.

**ENTER.** Logic command specifying the step number of the next command step to be executed in the current logic block. The ENTER command is equivalent to an unconditional GOTO command.

**\*ENTERED.** A switch used by the screen processing logic. This switch is *off* if the user has not entered data into a screen field, and is set *on* after the user enters data.

**Error.** A message type that informs the user of an error condition that must be corrected.

## **Glossary-4**



**EXIT.** Logic command that terminates the current logic block.

**EXTERNAL.** Logic command that passes control to an external program written in a language other than HP ALLBASE/4GL.

## **F**

**FIELD.** Logic command for altering the behavior or attributes of a field on the current screen.

**\*FIELDNO.** An alterable communication area field that contains the current screen field number. It is treated as a numeric.

**Field Specification.** A dictionary entry that defines the characteristics of a field. A field specification defines the editing and data validation characteristics of a field.

**FILE.** Logic command for specifying reading from, writing to, or deleting from a file.

**\*FILENAME.** Communication area field that contains the external name for the current or most recently accessed KSAM file, serial file, or HP TurboIMAGE/iX data set. This field is also used for dynamic naming of serial files. You can specify the external name for a file as \*FILENAME, and then move the appropriate name into \*FILENAME to specify the file to be accessed at run time.

**\*FIND.** FILE command argument used for locating a record on a KSAM data file or an HP TurboIMAGE/iX data set. The FILE \*FIND command positions the file pointer at the first record with a key value equal to or greater than the value in the key field of the record buffer, or the value specified by the \*KEY= argument.

**\*FIRST.** FILE command argument used for specifying that the first record on a KSAM file, serial file, or HP TurboIMAGE/iX data set is to be read.

**\*FUNCTION.** Alterable communication area field that contains the name of the function being executed (or blank if not currently executing a function).

**Function.** HP ALLBASE/4GL logic entity. A function is similar to a subroutine in a conventional language system. It consists of between 1 and 30 lines of logic commands.

## G

**Generate.** Validate and transform HP ALLBASE/4GL source parameters to executable format.

## H

**Hash.** The character #. This character is used in the MATH and MATHLOOP commands to specify that a number is to be treated as an absolute number.

**HP ALLBASE/SQL.** A relational database management system which operates on both the HP-UX and MPE/iX operating systems.

**HP TurboIMAGE/iX.** A network database management system which operates on the MPE/iX operating system.

## I

**IF.** Logic command for performing conditional tests.

**IFLOOP.** Logic command for repeated conditional testing.

**\*INDEX=.** Command argument for the CHECK, FILE, and VALIDATE commands. This argument specifies the index to be used to access the file for the particular file access. Using \*INDEX= with these commands does not change the current value in \*INDEXNO.

**\*INDEXNO.** Alterable communication area field that contains the number of the file index to be used for subsequent accesses of an application data file. This field is treated as numeric.

**\*INSERT.** FILE command argument used for specifying that the current contents of the file buffer are written to the file.

**\*IOSTATUS.** Alterable communication area field that contains the return status code from the data manager after accessing an application data file. This field contains the value zero if no error is encountered. If a file error does occur, the HP ALLBASE/4GL data manager writes a file error code to this field. This field is treated as numeric.

**items.** Individual, named, components of the application, which together make up the working application. Each separate name in a catalog list represents an application item.

## Glossary-6

## **J**

**\*JOINER=.** LINK and LINKLOOP command argument for specifying insertion of a character or string of characters between each field to be concatenated.

**Justification.** Alignment of data on a boundary or margin. HP ALLBASE/4GL provides automatic justification of data within the left and right boundaries (minimum and maximum length) of the field. The data may be justified on the first character position (left justified) or the last character position (right justified). Data may also be centered within the field, or no justification need be performed.

## **K**

**KEYS.** Logic command to display a set of function keys on the screen.

**\*KEY=.** CHECK, FILE, and VALIDATE command argument specifying the key value used to access KSAM files or HP TurboIMAGE/iX data sets.

## **L**

**\*LAST.** FILE command argument specifying that the last record in an indexed file is to be read.

**LENGTH.** Logic command for calculating the length of the contents of a field.

**LINK.** Logic command for concatenating nominated fields.

**LINKLOOP.** Logic command used for concatenating a number of consecutive fields.

**Literal.** A string of characters surrounded by quotes.

**Logic block.** The commands that make up a process or function. Each logic block contains from 1 to 30 lines of logic commands.

## **M**

**\*MAP.** CLEAR command argument that specifies that the screen or scratch-pad field position is to be mapped (within the specified screen or the scratch pad).

**MATH.** Logic command for performing an arithmetic operation.

**MATHLOOP.** Logic command for performing repeated arithmetic operations.

**Mess.** A message type that is displayed for information only and does not require a response from the user.

**MESSAGE.** Logic command for displaying a message on the screen.

**MODE.** Logic command used for specifying the names of the files that are being used in the current process. This command can only be used in a process. Files can be defined as being updated or locked to the current process. Files that are not defined as being updated are available for reference only. The MODE command remains in effect until the next process is executed.

**\*MODIFY.** FILE command argument specifying that the contents of the file buffer are to be used to update or change an existing record on a fixed length record serial file, KSAM file, or HP TurboIMAGE/iX data set. This record must be the last one read for that file.

**\*MOREREC.** A switch used in conjunction with variable length record serial files. If HP ALLBASE/4GL reads sufficient characters to fill the file record buffer without encountering a newline character when reading from a serial file with variable length records, the data manager sets the switch \*MOREREC *on*. If HP ALLBASE/4GL encounters a newline character before the buffer is filled, or the number of characters read exactly matches the length of the buffer, \*MOREREC is set *off*.

**MOVE.** Logic command for copying contents of one field (or literal) to another field. The MOVE command can also copy the contents of one buffer to another.

**MOVELOOP.** Logic command for repeated copying of one set of fields to another.

N

**\*NEWTIE.** An alterable communication area field that may be set to contain the next screen field number to be selected. Moving a value into \*NEWTIE performs the same function as the logic command TIE. This field is numeric.

## Glossary-8

**\*NEXT.** FILE command argument specifying that the next record on the file is to be read.

**NOTE.** Logic command that allows the insertion of a comment in a logic block. HP ALLBASE/4GL ignores NOTE command steps when it executes the logic block.

**\*NULL.** IF and IFLOOP command argument for testing the status of the null indicator variable associated with fields on record buffers for HP ALLBASE/SQL tables and select lists. Also used as a command argument for the MOVE command to set the indicator variables to the null status.

**\*NUMERIC.** IF and IFLOOP command argument used for checking if a field contains only numeric characters. (That is, 0 through 9, +, -.)

## **O**

**\*OFF.** IF and IFLOOP command argument used for testing if a switch is currently *off*.

**OFF.** Logic command that sets a switch to its *off* state.

**\*ON.** IF and IFLOOP command argument used for testing if a switch is currently *on*.

**ON.** Logic command that sets a switch to its *on* state.

## **P**

**\*P.** CLEAR command argument indicating that the field or fields to be cleared are scratch-pad fields. By itself, the \*P argument indicates that all scratch-pad fields are to be cleared.

**Pad character.** A specified character used to fill a field. Typically a space or 0.

**\*PAGELINE.** Alterable communication area field containing the number of the current print line on the current page of a report. This field is treated as numeric.

**\*PAGENO.** Alterable communication area field containing the current page number of a report. This field is treated as numeric.

**\*PASS.** Alterable communication area field used to pass arguments and results. For example, the CHECK command places the matching position

for data in a file or table into \*PASS. This field may be referenced by substring.

**\*PRESENT.** VALIDATE command argument used to check if the specified KSAM data file record field exists.

**\*PREVFLD.** Communication area field containing the number of the last screen field to be successfully committed.

**\*PREVIOUS.** FILE command argument specifying that the previous record on a fixed length record serial file, KSAM file, or HP TurboIMAGE/iX data set is to be read.

**PRINT.** Logic command to print a defined report line group from within a function that has been called during the generation of a report. The PRINT command is only available during report generation, and is ignored at any other time.

**PROCEED.** Logic command to execute a process logic block.

**Process.** An HP ALLBASE/4GL process is similar to a program in a conventional language system. Each process logic block contains from 1 to 30 lines of logic commands.

**\*PROCESS.** Alterable communication area field name containing the name of the current process.

## **Q**

**Query.** A message type that requires the user to enter a response.

## **R**

**Range.** A validation range specifying the lower and upper limits for the contents of a field.

**\*RANGE=.** Report generator record selection option that specifies the name of a HP ALLBASE/4GL range to be used for testing the contents of a specified field when selecting records for reporting.

**\*READ.** FILE command argument specifying that the record with the given key is to be read.

**\*RECNO.** Communication area field that contains the number of the record just read or written for a fixed length record serial data file or an HP

## **Glossary-10**

TurboIMAGE/iX data set. The value in \*RECNO is undefined after HP ALLBASE/4GL accesses any other data file type.

**\*REFRESH.** SHOW command argument specifying that the contents of the nominated fields are to be obtained from the main or default data movement field.

\*REFRESH is also a command argument for the EXTERNAL command. If \*REFRESH is specified, HP ALLBASE/4GL restores the terminal configurations and refreshes the screen on return from an external program.

**REPORT.** Logic command to execute a report.

**\*REPORT.** Alterable communication area field containing the name of the report being executed (or blank if not currently executing a report).

**\*RESET=.** DISPLAY command argument that clears all lines from the nominated line to the end of the scroll area before any further display within the scroll area.

**\*ROUTINE.** Alterable communication area field which contains the name of the current external routine (or blank if not currently in a called routine).

**\*ROWCOUNT.** Communication area field containing a number indicating the number of rows that HP ALLBASE/SQL processes as the result of a command that changes an SQL table.

## S

**\*S.** CLEAR command argument indicating that the field or fields to be cleared are screen fields. When used by itself, \*S indicates that all fields on the screen are to be cleared.

**Scratch Pad.** The HP ALLBASE/4GL scratch pad contains up to 99 scratch-pad fields. The scratch-pad fields are available for temporary storage of data. The scratch pad is dynamic and each scratch-pad field takes on the attributes of the data moved into it.

**\*SCREEN.** Alterable communication area field containing the name of the current screen (or blank if no screen).

**SCREEN.** Logic command to display a screen.

**SCROLL.** Logic command that displays data in the scroll area of a screen.

**SELECT.** Logic command that executes one of a series of commands from a selection list. The system executes the command identified by the value contained in the communication area field \*PASS.

**select list.** A “virtual” SQL table. The columns of a select list are taken directly from an existing table or view, or computed using an expression or aggregate function. HP ALLBASE/4GL automatically builds a record layout for each select list.

**SERIES.** Logic command that executes the specified command step number, or range of command steps in the current logic block. After the nominated commands have been executed, the command immediately following the SERIES command is then executed.

**SHOW.** Logic command that displays the current contents of a specified screen field or fields. When used with the \*REFRESH option, the SHOW command refreshes the screen buffer from the main or default data movement fields.

**\*SHOWING.** A switch used by the screen processing logic. When the SHOW command is displaying the contents of a field, \*SHOWING is set to *on*.

**SQL Logic Block.** Mechanism for passing SQL commands from HP ALLBASE/4GL to HP ALLBASE/SQL. An SQL logic block can contain up to eight SQL commands. These commands are passed to HP ALLBASE/SQL when the SQL logic block is invoked by an SQL command in an HP ALLBASE/4GL logic block.

**Subscript.** An integer or integers used to refer to a specific occurrence of a file record field when the field is specified as occurring more than once. HP ALLBASE/4GL subscripts are enclosed in parentheses.

**Substring.** A portion of the contents of a field. A substring is referenced by giving its starting character position in the field, a comma, and the length (in characters) of the substring, all enclosed in square brackets. For example, [3,2] is a substring of length 2, starting at character position number 3.

**\*SUITE.** Read-only communication area field containing the name of the application being run. If the current application is a version derived from a base application, \*SUITE contains the name of the base application.

## Glossary-12



## T

**\*TABLE=.** CHECK command argument that identifies the table against which checking is to be performed.

**TIE.** Logic command for specifying the next screen field to be selected. Used where the next screen field to be accessed is not necessarily the next field in sequence on the screen.

**\*TIME.** Read-only communication area field containing the current time. This field may be referenced by substring.

**TOP.** Logic command for specifying that step number 1 of the logic block is the next step to be executed.

**\*TOTALS(n).** A set of 16 alterable communication area fields used primarily in the report generator for totalling the values of numeric fields. The fields are identified as \*TOTALS(1) through \*TOTALS(16). These fields are treated as numeric.

**Training mode.** Operating mode used to prevent the updating of application data files from a specific user. While in training mode, all requests to update data files are ignored. The system administrator can set training mode *on* permanently for any user.

**TRANSACT.** Logic command that defines blocks of logically related file transaction commands.

## U

**UPDATE.** Logic command that updates all KSAM data files, serial data files, and HP TurboIMAGE/iX data sets whose buffer contents have been modified. This command has the same effect as issuing a FILE \*WRITE command for each relevant file.

**\*USER.** Read-only communication area field containing the name of the current user.

## V

**VALIDATE.** Logic command used to check if the contents of a specified field are missing (\*ABSENT) or present (\*PRESENT) on the relevant KSAM file record.

**Version.** An application that has been modified from a base application for a particular end user or group of end users.

**\*VERSION.** A read-only communication area field that contains the name of the current version being run.

**VISIT.** Logic command for executing a function logic block.

## **W**

**Warn.** A message type that beeps the terminal bell to alert the user, but does not require a response.

**WINDOW.** Logic command to display a window on the current screen.

**Window.** Screen that is overlaid on the current screen starting at the line number defined for the current screen. Window screens operate exactly as normal data screens. However, they may not contain windows or scroll areas. Window screens cannot be displayed independently of a normal data screen or on a menu screen.

**\*WRITE.** FILE command argument used for specifying that the contents of the file buffer are to be written to a serial file, KSAM file, or HP TurboIMAGE/iX data set.

## **X**

(no entry)

## **Y**

(no entry)

## **Z**

**ZIP.** Logic command that performs no operation and allows the next command to be executed.

# Index

---

## A

ABORT message, 7-67  
access class, SQL, 2-39  
action codes, 2-25  
action items on menus, 3-20  
administrator  
    application, 1-4  
    sign-on, A-2  
after function if blank, 3-40  
after selection functions, 11-15  
alphanumeric constants, 7-14  
    option, 9-12  
    product, 9-12  
    record, 7-14  
AND/OR connectives, 7-24, 7-26, 7-32,  
    7-34, 7-43, 7-45  
application  
    definition, A-5  
    definition (SQL), A-10  
    generation, 5-7  
    password, A-6  
    structure, 1-6  
    testing, 5-2  
    titles, 3-19, 7-6  
application title  
    application\_name, 7-7  
auto flow, screen fields, 3-38  
automatic numbering, data screen fields,  
    3-26

## B

behavioral characteristic, screen fields,  
    3-37  
block functions in screen painter, 3-15  
\*BYPASS switch, 6-14, 11-15

## C

calculated item  
    option\_cost, 11-37  
calculated item functions, 11-36  
calculated items, 7-6, 11-35  
case conversion, 2-11  
catalog display utility, 5-12  
class, access SQL, 2-39  
clear fields on commit, 3-26  
clearing fields on screens, 3-38  
column entry mode, in screen painter,  
    3-30  
command windows, 4-7  
commit actions  
    automatic, 3-38  
    field, 5-27  
    field commit, 9-34  
communication area fields  
    \*COUNT, 6-3  
    \*CROSS, 6-3, 6-26  
    \*FIELDNO, 5-22, 5-29, 9-34  
    \*NEWTIE, 5-23  
    \*ROWCOUNT, 10-60  
    \*SCREEN, 3-17  
    \*TIME, 3-18  
    \*TOTALS, 6-3, 6-26, 11-27, 11-31

**Index-1**

component printing utility, 5-16  
conditional tests, 7-24, 7-24, 7-32, 7-34,  
7-43, 7-45  
connectives, AND/OR, 7-24, 7-26, 7-32,  
7-34, 7-43, 7-45  
control breaks in reports, 11-11  
copying utility, 5-13  
\*COUNT communication area fields,  
6-3  
\*CROSS communication area fields,  
6-3, 6-26  
cursor keys, 2-7  
cursor position, in screen painter, 3-10

## D

D- actions, 3-4  
database definition, for SQL application,  
A-10  
database name, 2-46  
data fields on reports, 6-23  
data file manager, 2-26  
data files  
creation, 2-34  
default record, 2-32  
definition, 2-31  
error return codes, 7-11, 7-13, 9-12  
external name, 2-32  
file type codes, 2-32  
HP ALLBASE/4GL names, 2-32  
KSAM, 2-30  
multiple record layouts, 9-38  
option, 9-8, 9-9, 9-10  
order, 8-6  
product, 2-32  
record layout list, 2-32  
report primary file, 6-2  
data manager, 1-3  
data movement fields  
control of, 5-21  
default, 3-43  
definition, 3-42

other, 3-43  
primary, 3-41, 3-42  
data screens  
after function, 5-29  
after function if blank, 3-40  
after functions, 3-39  
auto flow, 3-38  
automatic field numbering, 3-26  
both functions, 3-39  
clearing fields, 3-26, 3-38  
data movement, 5-29, 5-30  
data validation, 5-28  
definition, 3-25  
display fields, 3-37  
field behavior specification, 3-37  
field commit action, 9-34  
field details, 3-34  
field display logic, 5-24  
field functions, 3-39  
field level help, 3-37  
field sequence number, 3-35  
generation, 3-44  
header, 3-25  
help screen name, 3-37  
include in SHOW, 3-38  
input field logic, 5-25  
input fields, 3-37  
painting, 3-28  
prior functions, 3-39  
processing logic, 5-21  
properties, 3-3  
required fields, 3-37  
scroll area, 9-18  
SHOW functions, 3-39  
system items, 3-29  
text items, 3-29  
window display, 9-22  
window starting line, 9-19  
data sets  
database name, 2-46  
default record, 2-46

## Index-2

- definition, 2-44
- file type codes, 2-45
- HP ALLBASE/4GL names, 2-45
- HP TurboIMAGE/iX, 2-44
- option, 9-10
- order, 8-7
- product, 2-45
- record layout list, 2-48

data validation

- data screen input, 5-28
- error messages, 5-28
- range, 7-74

date format, 3-18

date system item, 3-17

decision table

- prod\_opt\_update, 10-25

decision tables

- actions, 10-29
- generation, 10-37
- header screen, 10-25
- IF test, 10-28
- questions screen, 10-27
- relationships, 10-32
- usage, 10-24

DECLARE CURSOR command, 4-54

default data movement field, 3-43

default record layout, 2-32, 2-46

DEFINE command, 9-43

defining a process, 4-5

deleting items on screens, 3-13

deletions utility, 5-14

developer user name, A-4

developer validation, A-3

development security codes, A-7

dictionary

- application titles, 3-19
- field specifications, 2-8
- select list name, 4-54
- storage items, 7-5
- table name, 4-54

DISPLAY command, 9-46, 9-51, 9-55

display field processing, 5-24

display fields on screens, 3-31, 3-33, 3-37

displaying screens, 3-4

display listing utility, 5-12

display only fields on screens, 3-33

DM IMAGE \*LOCK command, 4-46

DM IMAGE \*MODE command, 4-23

DM IMAGE \*UNLOCK command, 4-30

duplicate keys, 2-23

**E**

editing logic blocks, 7-20, 7-30, 7-40

editing text, in screen painter, 3-11

\*ENDLINE switch, 6-14, 11-15, 11-24

ENTER command, 4-14, 4-21, 4-31

\*ENTERED switch, 5-29, 10-7, 10-9, 10-12

error message, 5-28

ERROR message, 7-67

errors during generation, 2-28

external file name, 2-32

**F**

FETCH command (SQL), 4-41

field commit actions, 5-27

field level help, data screens, 3-37

\*FIELDNO communication area, 5-22

\*FIELDNO communication area field, 5-29, 9-34

field specifications

- cost, 8-4
- definition screen, 2-8
- delivery\_date, 8-5
- description, 2-14
- forced uppercase, 2-11
- lead\_time, 2-15
- order\_date, 8-5
- order\_no, 8-3
- product\_no, 2-11

- quantity, 8-4
- report fields, 6-28
- screen field definition, 3-35
- screen fields, 3-31
- unit\_measure, 8-4
- FILE command
  - \*FIND argument, 9-46, 9-56
  - \*INDEXNO=, 4-38, 4-46
  - \*INSERT argument (SQL), 4-22
  - \*KEY=, 4-39, 4-46
  - \*NEXT argument, 9-46, 9-51, 9-56
  - \*NEXT argument (SQL), 4-41
  - usage, 4-13, 4-20, 4-29
- file linkages in reports, 11-19
- file manager, 7-11, 7-13, 9-12
- file name, external, 2-32
- files
  - creation, 2-34
  - HP TurboIMAGE/iX, 2-44
  - KSAM, 2-30
  - option, 9-8, 9-9, 9-10
  - order, 8-6
  - product, 2-32, 2-45
  - record references, 3-43
- \*FIND argument, FILE command, 9-46, 9-56
- forced case conversion, 2-11
- function details screen, 4-36
- function header screen, 4-35
- function keys
  - actions, 9-28
  - default definitions, 9-30
  - definition, 9-29
  - field commit from function, 5-27
  - label text, 9-31
  - product\_keys, 9-30
  - switches, 9-31
- functions
  - add\_option, 10-40, 10-41
  - add\_product, 7-59, 7-59
  - after, 3-39, 5-29
  - after print, 6-14
  - after selection, 11-15
  - before print, 6-14
  - both, 3-39
  - calculated item, 11-36
  - del\_all\_options, 10-51, 10-67
  - del\_corrupt\_prod, 10-54, 10-70
  - delete\_option, 10-56, 10-62, 10-72
  - delete\_product, 7-59, 7-62, 10-48, 10-58, 10-64
  - end of report, 6-9
  - generation, 4-49
  - get\_1st\_option, 9-45, 9-50, 9-55
  - get\_next\_option, 9-48, 9-53, 9-54, 9-57
  - modification, 7-20, 7-30, 7-40
  - modify\_option, 10-45, 10-46
  - modify\_product, 7-58, 7-61
  - new\_option, 10-14
  - option, 9-33
  - option\_cost, 11-37
  - option\_key\_read, 10-14, 10-17
  - option\_keys, 9-32
  - option\_no, 10-43, 10-44
  - option\_present, 11-23
  - print\_option, 11-23
  - prior, 3-39
  - prod\_add, 7-49
  - prod\_delete, 7-50
  - prod\_modify, 7-49
  - product, 9-34
  - product\_key\_read, 4-37, 4-41, 4-45, 7-50, 10-6, 10-11
  - product\_range, 11-6, 11-8
  - screen field, 3-39
  - scroll\_options, 9-42
  - sel\_product, 6-9
  - start of report, 6-9

## Index-4

## G

- generate program, 2-28
- generation
  - decision tables, 10-37
  - error messages, 2-28
  - functions, 4-49
  - log file, 5-19
  - menu, 5-18
  - messages, 7-69
  - processes, 4-32
  - record layouts, 2-28
  - reports, 6-31
  - resolution of references, 5-7
  - screens, 3-44
  - select lists, 9-41
  - SQL commands, 4-54
  - SQL logic blocks, 4-55

## H

- help screens
  - definition, 9-61
  - names, 3-37
  - product, 9-63
  - product\_no, 9-62
  - product\_no\_error, 9-63
- host variable referencing, (SQL), 4-55

## I

- IF command, 7-22, 7-32, 7-42
- include in SHOW, 3-38
- initial action
  - correcting mistakes, 5-9
  - name, A-8
  - type, A-8
- input fields on screens, 3-31, 3-37
- item names, 2-2

## J

- joining HP ALLBASE/SQL tables, 11-10

## K

- key fields, 2-25
- KSAM data manager, 1-3

## L

- layout keys, in screen painter, 3-13
- line drawing characters, 9-20
- line groups in reports, 6-2
- line headers in reports, 6-10
- line numbers in reports, 6-3
- linkages in reports, 11-19
- LINK command, 10-15, 10-20, 10-23
- LINKLOOP command, 10-40
- listing application components, 5-12
- local printers, 6-7
- log file, generate errors, 5-19
- log file, trace mode, 5-7
- logic block, 4-2
  - SQL, 4-50
- logic block modification, 7-18, 7-20, 7-30, 7-40
- logic blocks
  - SQL, 4-2
- logic commands
  - DEFINE, 9-43
  - DISPLAY, 9-46, 9-51, 9-55
  - DM IMAGE \*MODE, 4-23
  - DM IMAGE \*UNLOCK, 4-30
  - ENTER, 4-14, 4-21, 4-31
  - FILE, 4-13, 4-20, 4-29
  - IF, 7-22, 7-32, 7-42
  - LINK, 10-15, 10-20, 10-23
  - LINKLOOP, 10-40
  - MODE, 4-8, 4-15, 4-27
  - MOVELOOP, 11-7, 11-8
  - PROCEED, 7-50, 9-47, 9-52, 9-56
  - SCREEN, 3-4, 4-11, 4-18, 4-28
  - SCROLL, 9-44
  - SHOW, 4-40, 4-43, 4-48
  - TIE, 5-23
  - TRANSACT, 10-4, 10-23

- logic constructs
  - decision tables, 10-24
  - functions, 4-1
  - processes, 4-1
- logic step modification, 7-25, 7-37, 7-47

## **M**

- menus
  - action items, 3-2, 3-20
  - actions, 3-4
  - features, 3-2
  - headers, 3-7
  - menu bypass, 6-32
  - processing, 3-2
- messages
  - add\_option, 10-22
  - add\_prod, 7-69
  - all\_opt\_deleted, 10-60
  - construction, 7-69
  - corrupt\_prod\_del, 10-56, 10-72
  - definition, 7-69
  - delete\_option, 10-57, 10-63, 10-73
  - delete\_options, 10-53, 10-60, 10-69
  - delete\_prod, 10-51, 10-60, 10-67
  - del\_prod\_opt, 10-22
  - file\_error, 7-69
  - generation, 7-69
  - no\_option, 10-47
  - no\_product, 7-72
  - no\_transact, 10-5
  - opt\_add\_ok, 10-45
  - opt\_delete\_fail, 10-57, 10-64, 10-73
  - opt\_delete\_ok, 10-57, 10-63, 10-73
  - option\_warn, 10-45
  - opt\_modify\_ok, 10-47
  - opt\_not\_exist, 10-22
  - prod\_add\_fail, 7-72
  - prod\_add\_ok, 7-71
  - prod\_corrupt, 9-48, 9-53, 9-57
  - prod\_delete\_fail, 7-72
  - prod\_delete\_ok, 7-72

- prod\_modify\_ok, 7-72
- product\_no\_error, 7-77
- types, 7-67
- unit\_error, 9-16
- MESS message, 7-67
- MODE command, 4-8, 4-15, 4-27
- modifying a step, 7-25, 7-37, 7-47
- modifying logic blocks, 7-18, 7-20, 7-30, 7-40
- module
  - file maintenance, 8-18
- module builder
  - building process, 8-16
- modules
  - order, 8-17
- MOVELOOP command, 11-7, 11-8
- moving and copying items on screens, 3-14
- MPE/iX operating system, 1-3
- multiple record layouts, 9-38

## **N**

- naming rules, 2-2
- \*NEWTIE communication area field, 5-23
- \*NEXT argument, FILE command, 9-46, 9-51, 9-56
- numeric constants, 7-10
  - end\_of\_chain, 9-13
  - end\_of\_file, 7-13, 9-12
  - record\_not\_found, 7-11
  - zero, 9-12

## **O**

- object code, 1-5
- operating system, 1-3
- operating system interface, 1-3
- OR connective tests, 7-24, 7-24, 7-32, 7-34, 7-43, 7-45
- other data movement field, 3-43

## **Index-6**



## **P**

- pad character, variables, 7-8
- painting a window, 9-22
- painting data screens, 3-28
- painting menus, 3-9
- paper for reports, 6-7
- password, application, A-6
- password, developer, A-4
- perform on SHOW, 3-39
- primary data movement, automatic
  - default, 3-41
- primary data movement field, 3-42
- primary file for report, 6-2
- primary key fields, 2-26
- printers, 6-7
- printing utility, 5-16
- print line suppression, 6-14
- PROCEED command, 7-50, 9-47, 9-52, 9-56
- processes
  - definition, 4-5
  - generation, 4-32
  - logic constructs, 4-1
  - main, 9-48, 9-53, 9-57
  - mb\_order, 8-17
  - modification, 7-20, 7-30, 7-40
  - process details screen, 4-6
  - process header screen, 4-5
  - product\_proc, 4-5, 7-18, 7-29, 7-39, 10-2
- product\_proc process modifications, 7-18, 7-29, 7-39

## **Q**

- QUERY message, 7-67

## **R**

- record layout list, 2-48
- record layouts
  - definition, 2-18
  - details screen, 2-21

- duplicate keys, 2-23
- generation, 2-28
- header, 2-19
- key fields, 2-25
- multiple, 9-38
- option, 9-8, 9-9, 9-10
- option\_scroll, 9-38
- order, 8-6
- primary key, 2-26
- product\_rcrd, 2-20
- select list, 9-41
- record selection for reporting, 11-13
- record sorting for reports, 11-11
- report painter
  - data fields, 6-23
  - dictionary field specifications, 6-28
  - entry prompt, 6-18
  - field details, 6-25
  - menu path, 6-17
  - saving reports, 6-29
  - text literals, 6-21
  - window keys, 6-19
- reports
  - after print line function, 6-14
  - before print line function, 6-14
  - control breaks, 11-10, 11-11
  - counting print lines, 6-13
  - definition, 6-1
  - end of report function, 6-9
  - file linkages, 11-19
  - generation, 6-31
  - header screen, 6-6
  - line groups, 6-2, 6-11
  - line headers, 6-10
  - line numbers, 6-3, 6-11
  - manual record selection, 11-15
  - output file, 6-7
  - page size definition, 6-9
  - primary file, 6-2, 6-7
  - printers, 6-7
  - prod\_opt, 11-3

- product\_rept, 6-6
- record sorting, 11-11
- selection criteria, 11-13
- sort order, 11-10
- start up function, 6-9, 11-4
- stationery, 6-7
- suppressing print lines, 6-14
- totalling, 6-3
- totalling facilities, 11-27, 11-31
- user input, 6-9, 11-4
- required fields, 3-37
- \*ROWCOUNT communication area field, 10-60

## **S**

- saving a screen, 3-22
- scratch-pad fields, 7-6, 11-7, 11-8
- SCREEN command, 3-4, 4-11, 4-18, 4-28
- \*SCREEN communication area field, 3-17
- screen field details
  - definition screen, 3-34
  - introduction to, 3-34
- screen field functions
  - after, 3-39
  - both, 3-39
  - prior, 3-39
  - show, 3-39
  - usage, 3-39
- screen fields
  - data acceptance, 5-21
  - functions, 5-21
  - processing sequence, 5-21
- screen field sequence number, 3-35
- screen information display, 5-21
- screen painter
  - block functions, 3-15
  - column entry mode, 3-30
  - cursor movement, 3-12
  - cursor position, 3-10

- data fields, 3-31
- date fields, 3-17
- deleting items, 3-13
- display fields, 3-31, 3-33
- editing action items, 3-21
- entering text, 3-11
- entry prompt, 3-9
- field specifications, 3-31
- input fields, 3-31
- layout keys, 3-13
- line drawing characters, 9-20
- menu action items, 3-20
- menu path, 3-9
- moving and copying items, 3-14
- number of characters, 3-32
- overlay screen, 9-23
- painting a menu, 3-9
- painting data screens, 3-28
- painting windows, 9-22
- saving screens, 3-22
- screen name, 3-16
- special text, 9-20
- system items, 3-16
- time fields, 3-18
- screen processing
  - input field logic, 5-25
  - logic, 5-21
  - logic, field display, 5-24
  - screen processing cycle, 5-21
- screens
  - data, 3-3
  - main, 3-8
  - mb\_order, 8-17
  - menu header, 3-7
  - menus, 3-2
  - module builder, 8-8, 8-11
  - new\_option, 9-25
  - option, 9-22
  - product\_scrn, 3-26, 9-18
  - prod\_wnd, 9-26
  - select\_products, 11-5

## **Index-8**

- types, 3-2
- windows, 3-3
- scroll area on screens, 9-18
- SCROLL command, 9-44
- SELECT command limitations, 4-54
- SELECT command (SQL), 4-54, 10-19
- selecting records for reporting, 11-13
- selection criteria screen, 11-13
- select lists
  - definition, 9-39
  - details screen, 9-41
  - generation, 9-41
  - header screen, 9-40
  - in SELECT command, 9-53
  - joining tables, 11-10
  - optscrol, 9-40
  - prod\_opt, 11-3
  - record layout, 9-41
- sequence number, screen field, 3-35
- SHOW command
  - include field, 3-38
  - \*REFRESH argument, 4-40, 4-43, 4-48
- SHOW functions, 3-39
- slave printers, 6-7
- sorting in reports, 11-10
- source code, 1-5
- special text, on screens, 9-20
- SQL command generation, 4-54
- SQL commands
  - SELECT, 4-54, 10-19
- SQL cursors, 4-56
- SQL database
  - access authority, A-11
  - definition, A-10
  - name, A-8
- SQL data manipulation
  - delete without using a cursor, 10-62
  - deleting records, 7-62
  - inserting records, 4-20
  - modifying records, 7-61, 10-46
- SQL DECLARE CURSOR command, 4-54
- SQL FETCH command, 4-41
- SQL logic block, 4-50
- SQL logic blocks, 4-2
  - commit, 4-57
  - definition, 4-51
  - details screen, 4-53
  - find\_prod, 4-54, 7-54
  - generation, 4-55
  - get\_optscrol, 9-52
  - get\_prod\_opt, 11-10
  - header screen, 4-51
  - host variable references, 10-19
  - limitations, 4-57
  - modify\_option, 10-46
  - modify\_product, 7-61
  - multiple commands, 10-62
  - opt\_del, 10-61
  - option\_key\_sel, 10-19
  - opt\_sel, 10-61
  - prod\_del, 7-63, 10-62
  - select\_product, 6-10
- SQL owner group, A-8
- SQL table
  - creation, 2-40
  - definition, 2-37
  - format, 2-42
  - ownership, 2-42
- SQL tables
  - option, 9-9
  - order, 8-7
  - product, 2-38
- start of report function, 6-9
- stationery for reports, 6-7
- step modification, 7-25, 7-37, 7-47
- storage items, 7-5
  - alphanumeric constants, 7-14
  - application titles, 7-6
  - calculated items, 11-35
  - numeric variables, 7-10

- scratch-pad fields, 11-7, 11-8
- variables, 7-8
- switches
  - \*BYPASS, 6-14, 11-15
  - \*ENDLINE, 6-14, 11-15, 11-24
  - \*ENTERED, 3-40, 5-29, 10-7, 10-9, 10-12
  - function key, 9-31
  - \*SHOWING, 3-40
- system items on screens, 3-16

## T

- tabbing sequence, 2-7
- table creation, SQL, 2-40
- terminal keys, 2-7
- testing, 5-2
- tests, connective, 7-24, 7-24, 7-32, 7-34, 7-43, 7-45
- text entry, in screen painter, 3-11
- text items on reports, 6-21
- TIE command, 5-23
- \*TIME communication area field, 3-18
- time fields, on screens, 3-18
- titles, application, 3-19, 7-6
- \*TOTALS communication area fields, 6-3, 6-26, 11-27, 11-31
- totals in reports, 6-3, 11-27, 11-31
- trace log file, 5-7
- trace mode, 5-4
- TRANSACTION command, 10-4, 10-23

## U

- undefined items, 5-7
- uppercase forced, 2-11
- users
  - developer names, A-4

- user input for reports, 11-4
- utilities
  - catalog display, 5-12
  - copying, 5-13
  - deletions, 5-14
  - generate menu, 5-18
  - menu, 5-10
  - printing, 5-16

## V

- validation
  - validation range definition, 7-74
- variables, 7-5, 7-8
  - confirm, 9-11
  - current\_record, 9-11
  - first\_product, 11-8
  - last\_product, 11-8
  - mode, 7-9
  - option\_no, 9-11
  - option\_status, 9-11
  - product\_status, 7-8
- vertically aligned prompts on screens, 3-30

## W

- WARN message, 7-67
- window
  - definition, 9-22
  - display, 9-22
  - screens, 3-3
  - starting line, 9-19
- windows
  - new\_option, 9-25
  - option, 9-22
  - prod\_wnd, 9-26

## Index-10