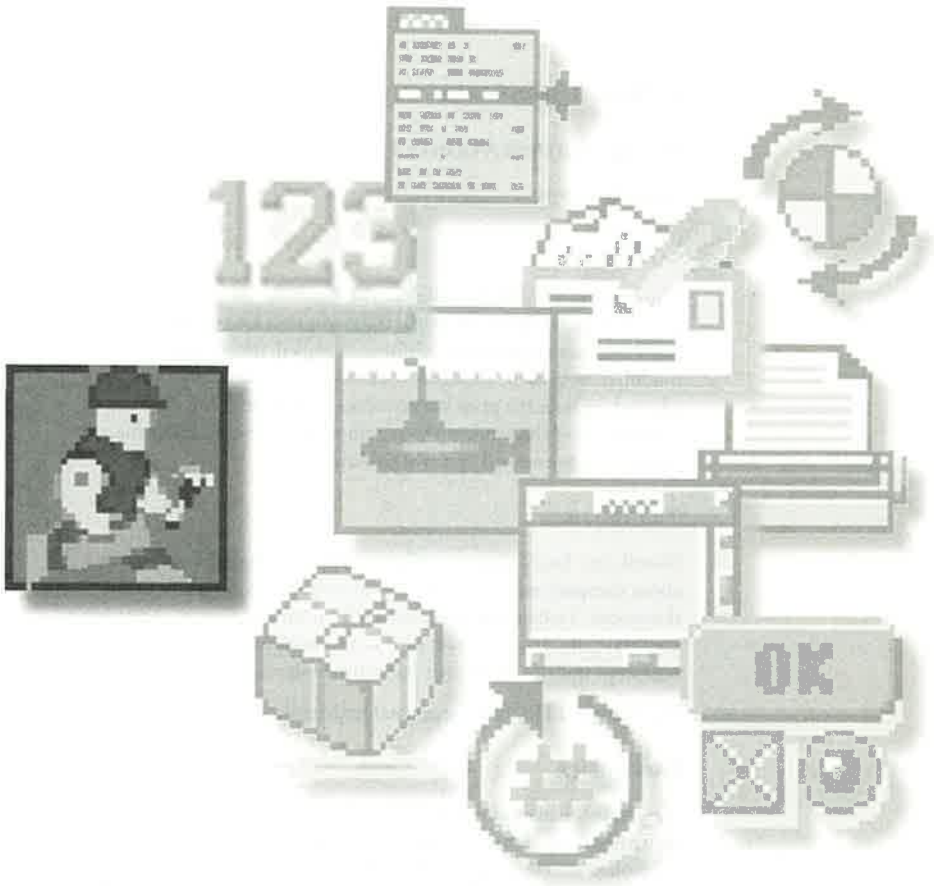
User's Guide

NOVELL®

for WINDOWS

AppWare™

APPLICATION DEVELOPMENT TOOLS



AppWare User's Guide

for MS Windows

NOVELL

AppWare User's Guide

Second Edition (version 1.1)

November, 1994

Documentation Team

Ed Firmage, Tom Hawkes, Kathy Jensen, Erik Nyström, and Susan Piele.

Part Number 100-002638-001

CONTENTS

CHAPTER 2 - TUTORIAL

APPENDIX C - KEYBOARD SHORTCUTS

Installation

INSTALLING APPWARE

Package Checklist

Your AppWare package for MS-Windows should contain:

✓ Disks

5 Install Disks (AppWare)

2 Install Disks (3rd Party ALMs)

✓ Guidebooks

ALM Builder Reference Guide

Attachmate ALMs

Brookhill ALMs

Cheyenne Software ALMs

Clear Access ALMs

Crystal Services ALMs

Essentials ALMs

ImPower ALMs

Parallel Software ALMs

Power ALMs

Techgnosis ALMs

AppWare User's Guide (this manual)

✓ License Agreements

Prototype

Limited Use

System Requirements

System Software

Minimum:

MS-Windows 3.0 or later

System Hardware

Minimum:

80386 PC

4 MB RAM

Recommended:

80486 PC

8 MB RAM

System Disk Space:

23 MB (AppWare & ALM Builder)

13MB (AppWare only)

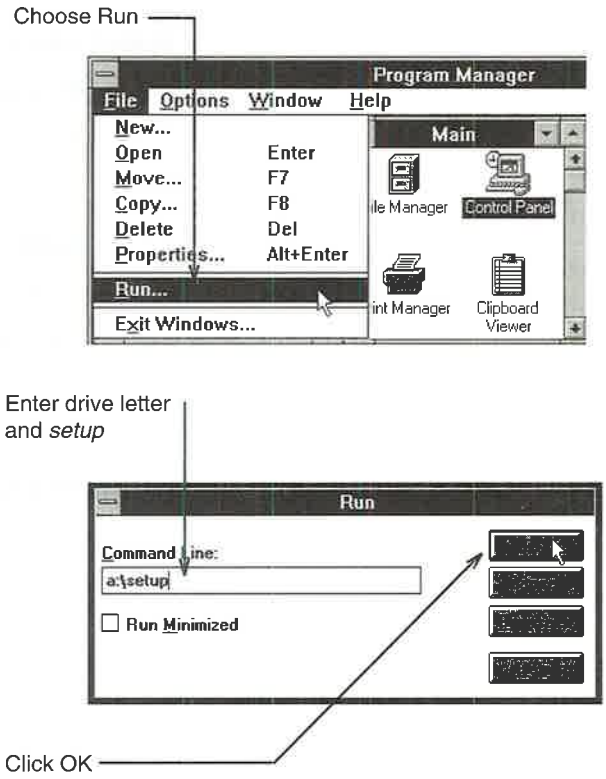
Installation Instructions

Before continuing, take a minute to make backup copies of your master disks. For instructions on copying disks, refer to your MS-DOS or Microsoft Windows documentation.

To install AppWare:

1. **Start Microsoft Windows if it is not currently running.**
2. **Insert the AppWare Install Disk #1 into your floppy drive.**
3. **Choose Run from the Program Manager's File menu.**
4. **In the dialog, type the letter of the floppy drive you're using followed by a colon (:), backslash (\), and the word "setup".**
5. **Click OK to start installing AppWare.**

   Follow the instructions as they appear on the screen.



Installation Summary

During installation, you specify the directory in which AppWare will be installed. The default installation directory is "\APPWARE". Inside the directory you specify, several sub-directories are automatically created.

The two sub-directories named "\BIN\DEGUG" and "\BIN\NODEBUG" are for AppWare and the ALM files containing the code that makes objects operate at run time. There is generally one ALM file per type of object. The "\BIN\DEBUG" directory contains debugging files used by the ALM Builder Utility, which is part of ALM Builder.

The third directory contains *.CFG or configuration files which hold the object and function icons, parameter information, and other settings which are used only by the AppWare environment at design time. The *.CFG files are not accessed by the built applications at run time.

The other sub-directories contain "Help" files and sample AppWare "Projects". These sub-directories may also contain supplementary files for the AppWare application.

WELCOME TO APPWARE

About The User's Guide

If you're one of those users who never read manuals, you'll no doubt want to jump right into the program. While we don't want to dampen your enthusiasm, we recommend that you take some time to familiarize yourself with this manual. You'll be more effective if you do. The next page tells you about the Guide chapter by chapter.

WHAT'S IN THIS GUIDE

1 – Introduction Provides a brief historical overview of approaches to computer programming and compares the AppWare environment with other approaches to application development.

2 – Tutorial Takes you through the processes of creating your own working application using AppWare.

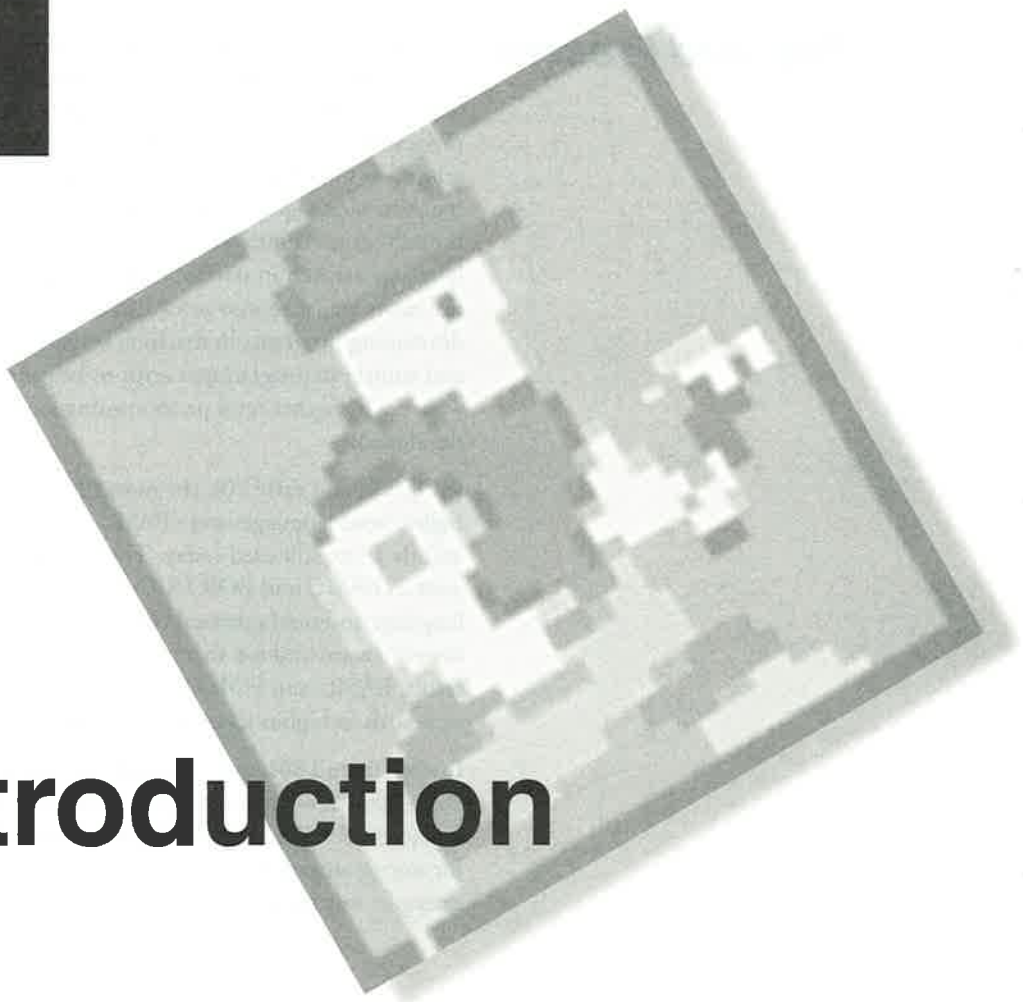
3 – Using AppWare Provides detailed topical explanations of the AppWare environment, including debugging and compilation.

A – Glossary Defines terms used in this manual.

B – Menu Reference Describes each menu item in AppWare.

C – Keyboard Shortcuts Lists keyboard shortcuts in AppWare.

1



Introduction

SOFTWARE DEVELOPMENT

Past and Present

Your PC, like all computers, can only process instructions and data expressed in binary form — some combination of zeros and ones. The most direct approach to programming is therefore to work in "machine language" or something close to it. The problem with machine language programming, however, is that humans don't think like machines. Our thought and language are rich in symbolic representation that does not easily translate into zeros and ones. As a result, developing programs in machine language is too tedious and subject to programmer error to be generally useful. That's why higher-level programming languages have developed.

In the 60s and early 70s, the most prominent of these higher-level languages were BASIC and FORTRAN, and these are still used today. The syntax and symbols used in BASIC and FORTRAN are based on English-language and mathematical conventions and are thus easier to learn and use than machine language. As a result, BASIC and FORTRAN have become worldwide standards in higher-level programming.

In the 70s and 80s, new, structural languages, such as Pascal and C, were introduced. These have largely replaced BASIC and FORTRAN as development tools for microcomputers such as your PC. AppWare, for example, was largely created in Pascal and C.

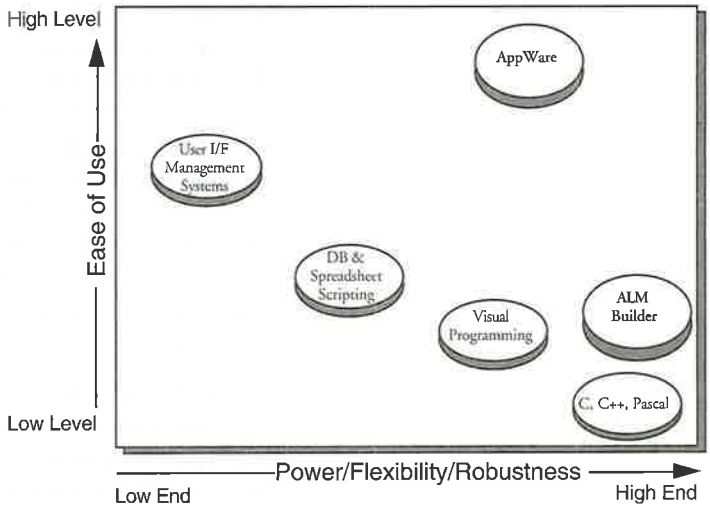
Unfortunately, the growth of the personal computer software industry during the 80s has outstripped the capacity even of these higher-level languages to meet the needs of a software-hungry market, and so another generation of tools had to be invented. While operating at the same level as Pascal and C, these new "object oriented" programming (OOP) languages approach the problem of application development a little differently. They are based on the idea of reusing programming code. To make reuse possible, code is written in the form of independent segments or objects. The idea of object programming is so potent that hardware and system vendors such as Apple Computer and IBM are now working to create object oriented operating systems. These will facilitate application development by making ready-made code segments accessible to software developers.

Despite these advances, the need for powerful, customized applications continues to outpace the capacity of the programming industry. The reason for this is that conventional programming tools, even higher-level languages like Pascal and C, or object oriented languages like C++, still involve the programmer in operating system level problems. Pascal, C, C++ and comparable languages are high-level relative to binary code or even assembly language, but operate far below the threshold of the enduser, who interacts with the computer at its highest level. From the viewpoint of the end-user, these languages are low-level tools.

As operating systems become more complex, the expertise required of software developers increases. So does the time required to create applications to exploit these operating systems. No fundamental change in the pace of software development can occur until there is a significantly higher level of application development — until endusers have the power to create their own software. That's what AppWare is all about.

The following diagram illustrates how we think AppWare compares with other kinds of development tools in terms of ease of use and programming power.

Application Development Environments



Low-level Programming Languages

C, C++, Pascal and other low-level languages offer great generality, power, and flexibility at the cost of ease of use. Users of such low-level tools must have an expert knowledge of the operating system for which they are developing applications. Applications developed in such tools are difficult to maintain and are generally not directly portable between platforms because of their operating system dependence.

Also in this category are low-level, graphic programming tools. Like more traditional programming languages, these inherit the strengths and weaknesses of all tools that work this closely with the operating system. They differ from the traditional tools primarily in their choice of a graphic rather than a text-based user interface.

Scripting Languages

These simplify the programming task by providing a higher-level script language translatable into executable code. Often incorporated into end-user applications such as spreadsheets, such languages may be limited by the scope of the associated application. Applications developed in these environments generally cannot operate without the application they were developed in, and thus involve increased cost and overhead.

GUI Builders

Products in this category facilitate application development by enabling you interactively and graphically to build a user interface. These systems then output the source code necessary to build the interface. The "meat" of the application underlying the interface must be separately developed in a low-level environment, with its inherent drawbacks relative to turn-around time, expense, ease of maintenance, and portability.

AppWare

AppWare raises the level of application development away from coding altogether, enabling programmers and nonprogrammers alike to create professional software with unprecedented speed. AppWare uses a comprehensive library of robust, high-level objects and functions that facilitate development without significant sacrifices in either performance or functionality. Using these objects is an order of magnitude less complicated than developing them — something that can't be said of objects in more conventional OOP environments. In other words, to use an object, you don't need to know anything about how it works at the code level.

ALM Libraries

The AppWare ALMs for Microsoft Windows contain dozens of high-level objects and hundreds of high-level functions divided among several object sets. The first of these is the Essentials ALMs, a core set of objects and functions supporting various aspects of the Microsoft Windows graphical user interface such as menus, lists, buttons, and windows, as well as traditional programming features such as loops, subroutines, and arrays — all structured so as to be usable by the non-programmer.

Other object sets address specialized areas of application development.

These include:

- Multimedia ALM — digitized sound and movies.
- Communications ALMs — for modem and serial connection, file transfer, and terminal emulation.
- Application Linking ALMs — for interapplication communications as well as interobject communication across networks. Included in this set are ALMs supporting OLE, DLL, and DDE.

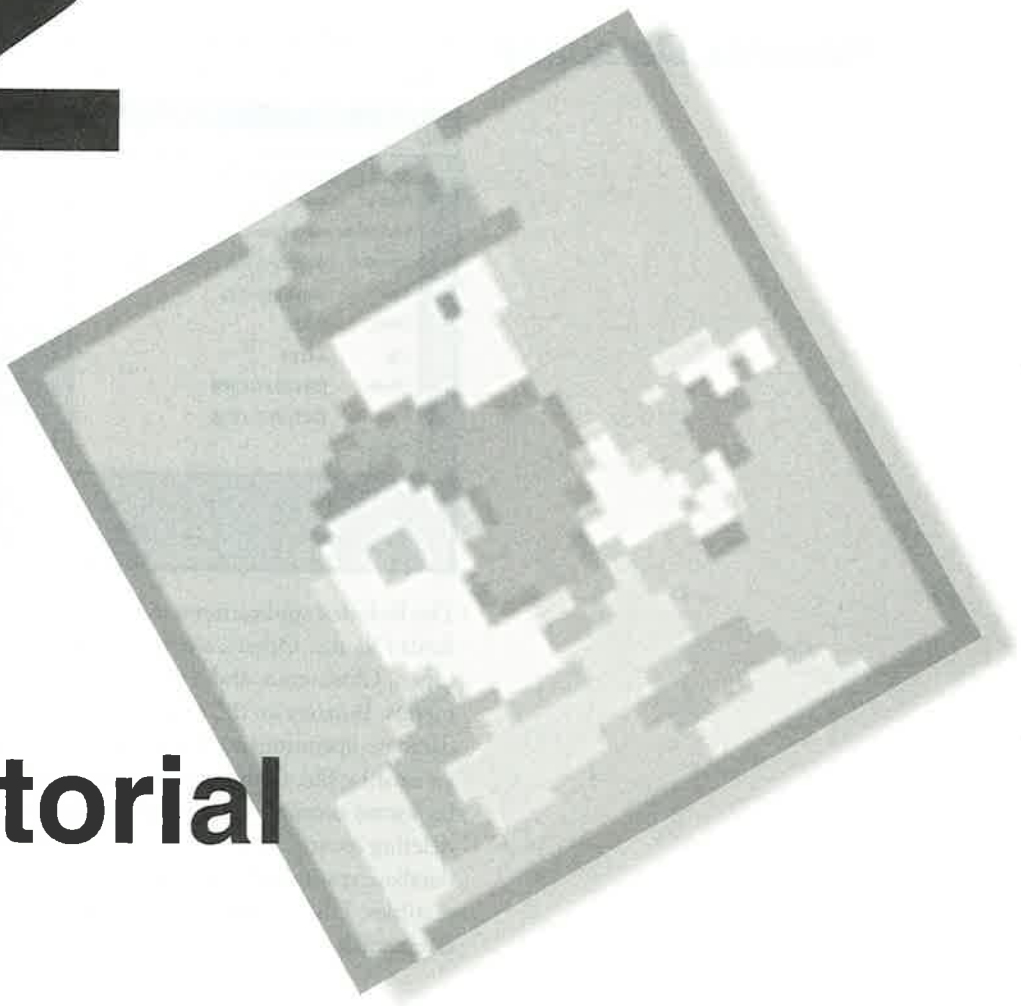
ALM Construction Kit

A set of tools that allows experienced programmers to create new ALMs for AppWare through the use of lower-level tools such as Borland C++, Microsoft C++, and Turbo Pascal for Windows.

Third Party ALMs

AppWare also includes a number of ALMs developed by third parties. For an up-to-date listing of these and other ALMs, consult the AppWare Solutions Guide, available if you call 1-800-277-2717.

2

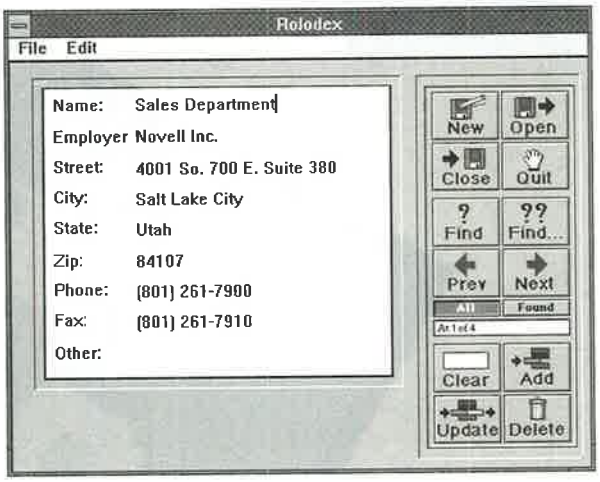


Tutorial

INTRODUCTION

Rolodex Database

In this chapter, you'll learn how to use AppWare by building the simple database program shown here.



The Rolodex application window contains nine data fields (Name...Other), a panel of control buttons (New, Open, Close, etc.), and a menu bar with File and Edit menus. Buttons on the control panel manage basic database operations such as creating a new file, opening an existing file, finding records, browsing forward and backward through the file, adding, updating, and deleting records. The File menu contains two additional database commands for importing and exporting ASCII text files. Edit includes standard Cut, Copy, and Paste commands.

As databases go, this one is rather basic. Once you're familiar with AppWare and its objects, you should be able to recreate this application in well under an hour.

Creating this program involves the following steps.

1. Building a menu bar. For this task, you'll work with the Menu Bar, Menu, and Menu Item objects.

   Your menu bar will contain a File menu with commands for importing and exporting records and for exiting, and an Edit menu with commands for Cut, Copy, and Paste.

2. Designing an application window for data fields and control buttons. For this task, you'll use the Window, Text, Database, and Browser objects.

3. Compiling and running your database program.

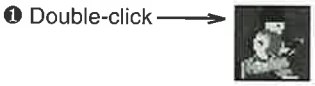
LESSON 1 BUILDING A BASIC DATABASE

Topics

Starting AppWare
Creating a Project
Creating a Subject
AppWare Interface
What is an Object?
Menu System
Creating Menus
Titling Object Groups
Titling Objects
Rearranging Objects
Editing Menu Items
Linking Menu Items to the File Menu
Linking Menu Items to the Edit Menu
Linking Menus to a Menu Bar
Shortcut for Editing Objects
What is a Function?
Using the Quit Function
Using Cut, Copy, & Paste Functions
Saving Your Project
Creating More Subjects
More Objects
Adding More Objects
Setting Up the Window
Layout Grid
Getting Information on Objects
Positioning Objects
Adding Static Text Labels
Drawing Boxes
Setting Up the Browser
The Database Object
Setting Up the Database Object
Setting Up the Text Objects
Sharing and Aliasing Objects
Adding a Menu Bar to the Window
Aliasing the Edit Menu
Adding Function Chains
The Browser & Function Signals
What is a Parameter?
Disabling the Edit Menu
Clearing Fields
Activating Text Field
Compiling & Running Your Application
Using the Rolodex
Delivering Compiled Applications

Starting AppWare

In this lesson, you'll set up a custom menu bar containing a File menu that displays a Quit (Exit) item and an Edit menu that displays items for Cut, Copy, and Paste. You'll design an application window to hold data fields, Browser control buttons, and background graphics. When you're done designing the sample database program, you'll compile and run it.

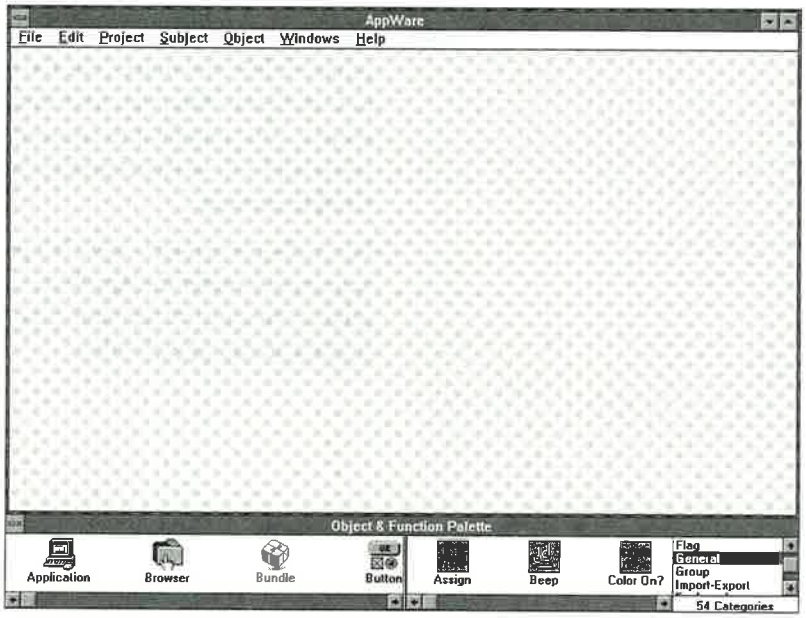


AppWare

To launch AppWare:

1. **Double-click the AppWare icon.**

   You'll see the AppWare menu bar at the top of the screen and Object & Function Palette across the bottom.



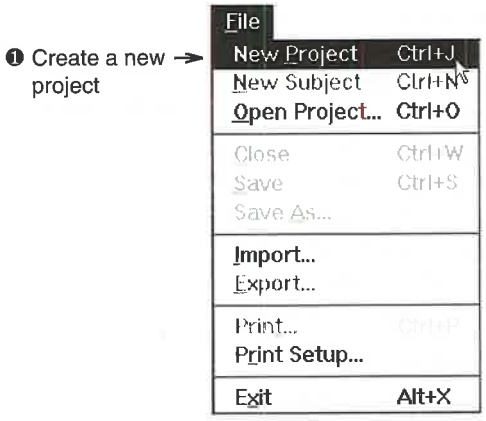
Creating a Project

Once you're in AppWare, the first step in building a new application is to create a project. This contains your application "code", organized in subprojects called *subjects*.

To create a project:

1. **Choose New Project from the File menu.**

   You should see an empty project window called Project1 appear at the top of the screen just below the menu bar. You can rename the project when you save it.



☛ **Note** Although much easier to create and understand, a project is like the source code file a programmer creates in C or Pascal. After you have finished constructing your project, you compile it to create an application, just as you would a C or Pascal project that can be accessed by Double-clicking.

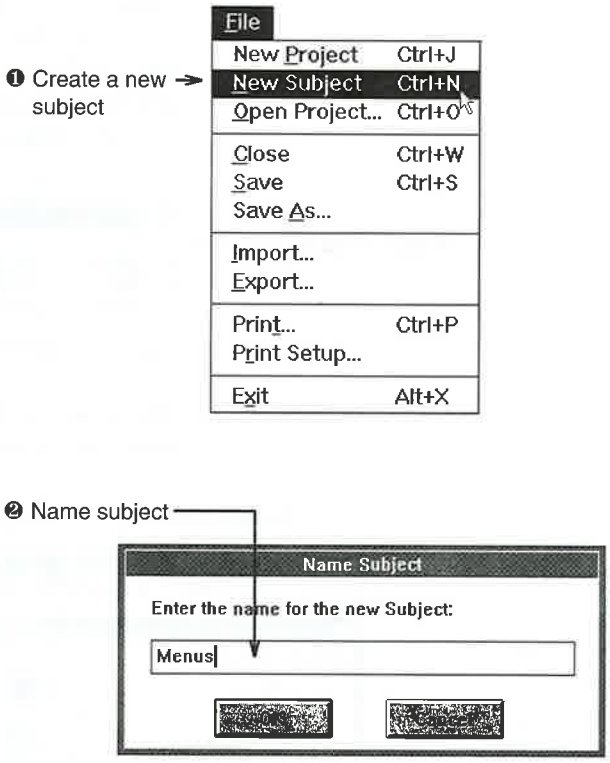
For each open project, there is a project window that appears just under the AppWare menu bar. If multiple projects are open at the same time, their windows are stacked on top of each other.

Creating a Subject

A project is made up of one or more reusable subprojects or subjects, each devoted to a different aspect of your application. You must have at least one subject in a project, so create a subject now. The subject you create here will contain objects for a menu bar.

1. **Choose New Subject from the File menu.**
2. **Type "Menus" when the program asks you for the name of the subject.**
3. **Click OK.**

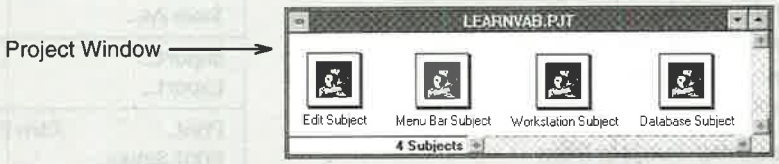
   Your Menus subject window appears in the middle of the screen. Its name, PROJECT1:Menus, indicates that it belongs to the project you created earlier. You'll rename the project later.



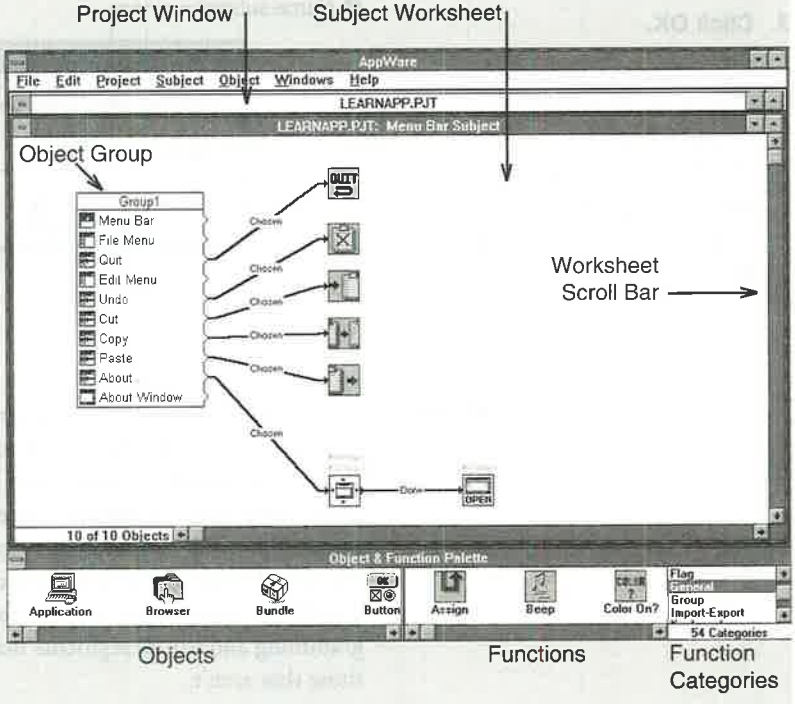
☛ **Note** Some subjects are so basic in nature that you'll reuse them time and again. For instance, you may create a basic menu bar subject containing File and Edit menus. You'll need a subject like this in just about every application. Other subjects may be very specific in nature — the sort of thing you may never reuse. General or specific, subjects let you compartmentalize your programming and isolate segments that are reusable from those that aren't.

AppWare Interface

Once you create a subject, a subject tile is deposited in the project window. The tile is labeled with the subject name. Double-clicking this tile brings up the subject window. Within the project window, you name and move subject tiles. You can also cut, copy, and paste subjects by selecting their tiles and choosing the appropriate menu item from the Edit menu.



The subject worksheet is used to contain objects and functions you drag from the Object & Function Palette.



What is an Object?

Objects are self-contained, reusable program building blocks containing both data and the code your computer needs to manage that data. You use a Text object, for example, to store and display textual data. Pictures are stored and displayed in Picture objects, lists in List objects, and so on.

Most AppWare objects represent application features that you'll immediately recognize. The Window object, for example, enables your application to display and operate windows. In this and most other cases, there's a one-to-one correspondence between a typical application feature (such as windows) and an AppWare object (such as the Window object).

Perhaps the most important characteristic of AppWare objects is that they are very high-level, meaning that they are easy to use. You don't need to be a programmer to know how to use an AppWare object. Nor do you need to know a lot about the inner workings of DOS or Windows. In fact, the code that makes an AppWare object work is invisible. You'll never see a line of it. To use an object, all you really need to know is how to use the application feature it represents. If, for example, you're familiar with the terms menu bar, menu, and menu item, and if you know how a Windows menu bar usually works, then you can build your own custom menu bar in AppWare.

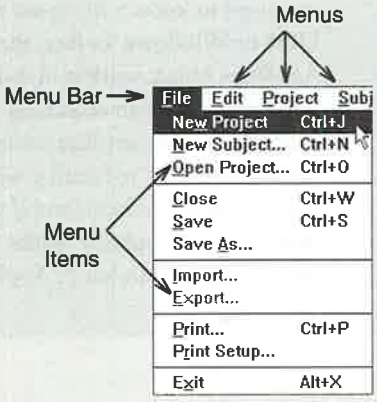
Menus

A menu system has three basic parts.

The first is the menu bar, which holds the menus displayed across the top of the window under the title bar.

Making up the menu bar are menus (File, Edit, etc.), containing topically arranged commands (New, Open, Close, Exit, etc.) that you choose to perform operations. By convention, File and Edit are the first and second menus in a menu bar.

Menu items are the menu commands you select to perform specific operations. While you're free to design your menus entirely as you wish, file-related commands such as New, Open, Close, Save, Print, and Exit are typically found in the File menu; editing commands such as Cut, Copy, and Paste are usually in the Edit menu.



Creating Menus

Now that you've created a subject, you can add objects to it. You do this by dragging them from the Object & Function Palette into the subject window. Objects that you drag into the subject window become a part of your application. Let's begin with the menu objects.

In AppWare, the menu bar, menus, and menu items are separate objects.

To create a menu bar:

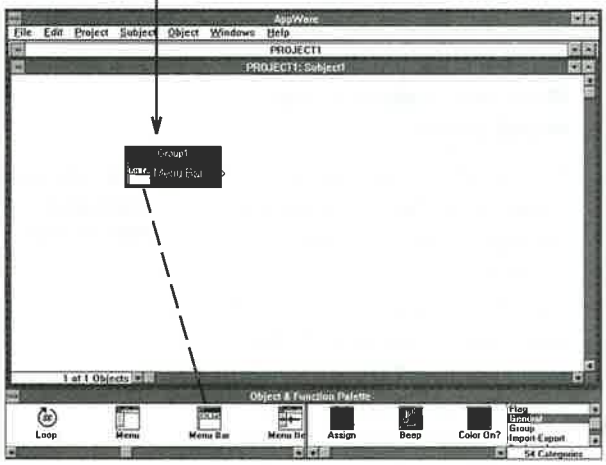
1. **Find the Menu Bar object in the objects compartment of the Object & Function Palette.**

   You may need to scroll through the palette to find the icon. Objects are displayed in alphabetical order.

2. **Drag the Menu Bar object onto the subject worksheet.**

This creates an object group with a copy of the Menu Bar object.

❷ Drag object onto worksheet

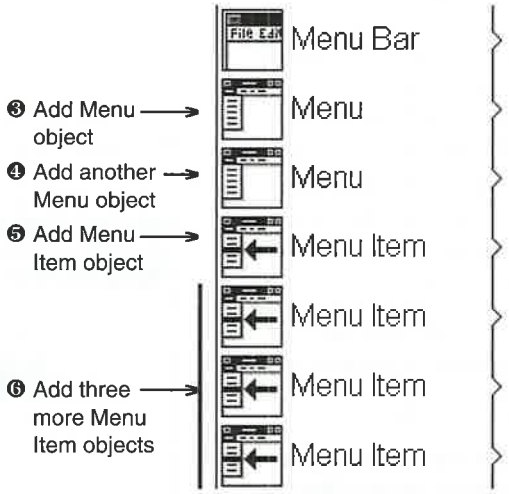


continued...

☛ **Note** When you drag an object into the subject worksheet, AppWare creates an object group. A group is simply a window-like container that holds one or more objects. You move groups, as you do windows, by clicking on the title bar and dragging with the mouse.

3. **Drag the Menu object into the object group that contains the Menu Bar object.**
4. **Repeat Step 3 so that you have two Menu objects in the subject.**
5. **Drag the Menu Item object into the object group.**
6. **Repeat Step 5 three more times so that you have four Menu Item objects in the object group.**

   You can also create multiple copies of an object by using copy and paste. To copy an object, select it and choose Copy from the Edit menu (or press Ctrl+C), then choose Paste from the Edit menu (or press Ctrl+V). If only one group is selected, the object is appended to the bottom of the group. If no group is selected, a new group is created and the object is pasted into it. If two or more groups are selected, the object is appended to the botom of the last group selected



☛ **Note** AppWare objects can be added to the worksheet or selected group by double-clicking

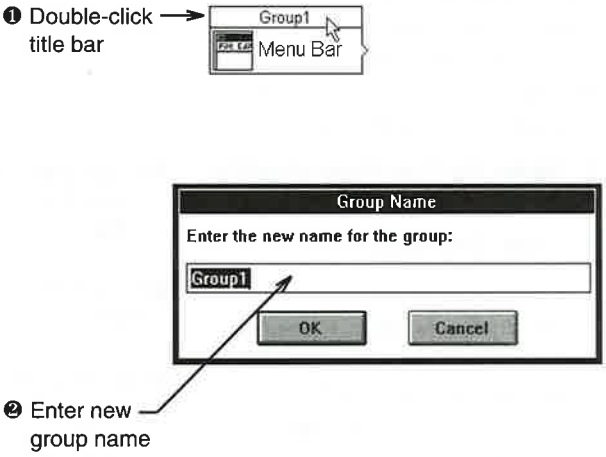
You have now added all the objects you need to incorporate File and Edit menus in your menu bar.

The next thing you'll do is name the object group and its objects.

Titling Object Groups

Giving names to your object groups is easy:

1. **Double-click the title bar of the object group you wish to rename.**
2. **Type the desired name in the dialog box. In this example, enter the name "Menu Group."**



Titling Objects

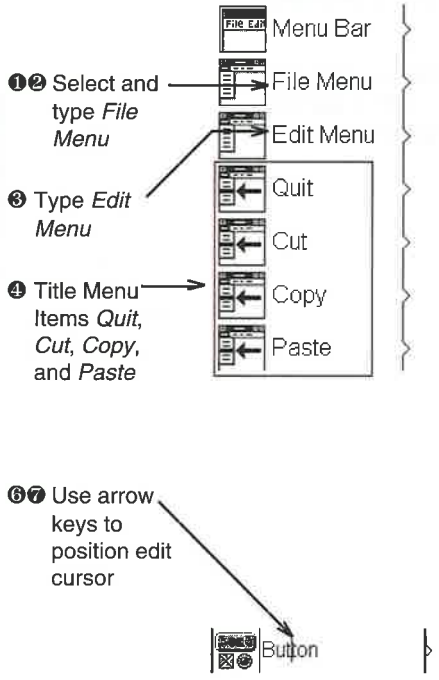
Now you'll title your Menu objects to help you distinguish them from one another. The titles you give objects in the object group are distinct from the names, if any, that the objects present to the user of your application at run time.

1. **Select the first Menu object in the object group.**
2. **Type "File Menu."**

   You should see the title of the Menu object change as you type. Click the object only once; double-clicking is used to display the object's editing dialog.
3. **Title the second Menu object "Edit Menu" as in step 1.**
4. **Title the Menu Items "Quit" "Cut," "Copy," and "Paste."**

To edit an object's title:

5. **Select the object.**
6. **Press the left or right arrow key.**
7. **Use the mouse or arrow keys to move the cursor in the object title and begin typing.**

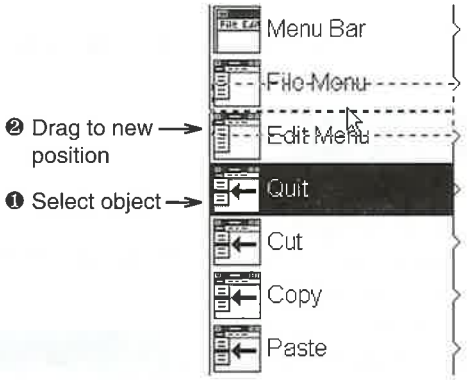


☛ **Note** The titles you have assigned here are labels that help you distinguish one object from another. However, these titles also become the default names of the menus and menu items at run time.

You will change the runtime names in each object's editing dialog later in this tutorial.

Rearranging Objects

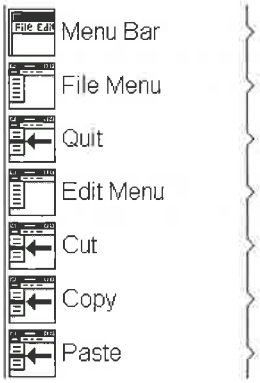
Once you have several objects in the object group, you will want to be able to identify which objects belong together. One way of doing this is to place related objects next to one another. In this case, let's put Quit underneath File Menu and the Cut, Copy, and Paste items under Edit Menu.



To rearrange an object in the object group:

1. **Select the object.**
2. **Drag the object to its new location in the object group.**

   Take a minute to rearrange your object group to look like the one illustrated here.



The arrangement of objects in the object group does not affect program operation. Like the group and object titles, the arrangement of objects in the group is an organizational convenience for you, the developer.

Editing Menu Items

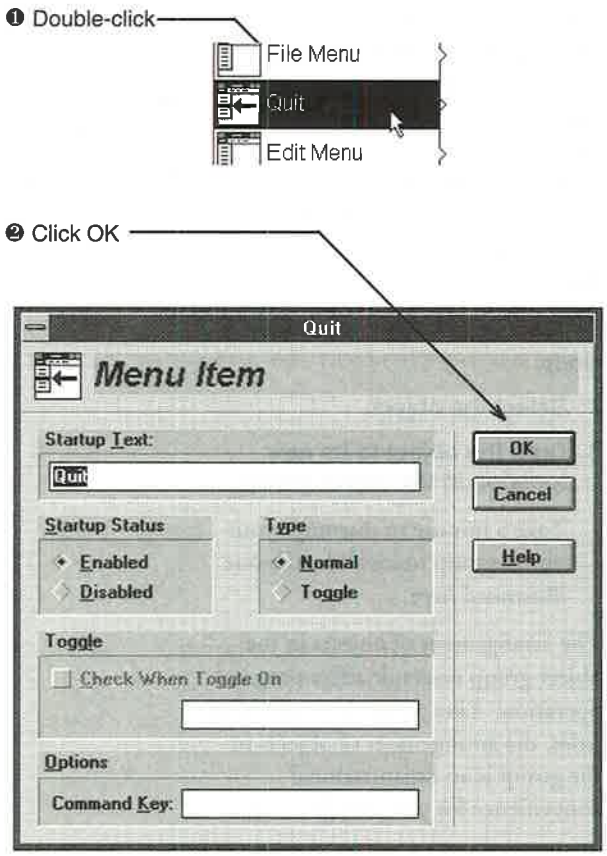
Next, you'll need to edit the objects themselves.

1. **Double-click the Quit Menu Item object in the object group.**

   The Menu Item editing dialog appears, allowing you to specify characteristics of the object. Notice that the title you gave the object appears in the Startup Text field. The Startup Text field contains the text that appears as the menu item's title at runtime. If the title is not correct, you can edit it here. No settings in this editing dialog need to be changed for this example.

2. **Click OK.**

You do not need to edit the other menu items (Cut, Copy, and Paste) at this time.



☛ **Note** Also, you can quickly go to the editing dialog by highlighting the object and pressing Enter.

Linking Menu Items to the File Menu

Now you need to put the menu items in the File and Edit Menu objects.

1. **Double-click the File Menu object in the object group.**

   The Menu object's editing dialog appears, allowing you to specify characteristics of the object. This dialog displays two lists. The left list, Objects in Menu, shows the Menu Item objects in the menu. At the moment, there are none. The right list, Available Objects, displays Menu Item objects you can add to the menu. You should see Quit, Cut, Copy, and Paste in this list.

2. **Select Quit in the Available Objects list.**

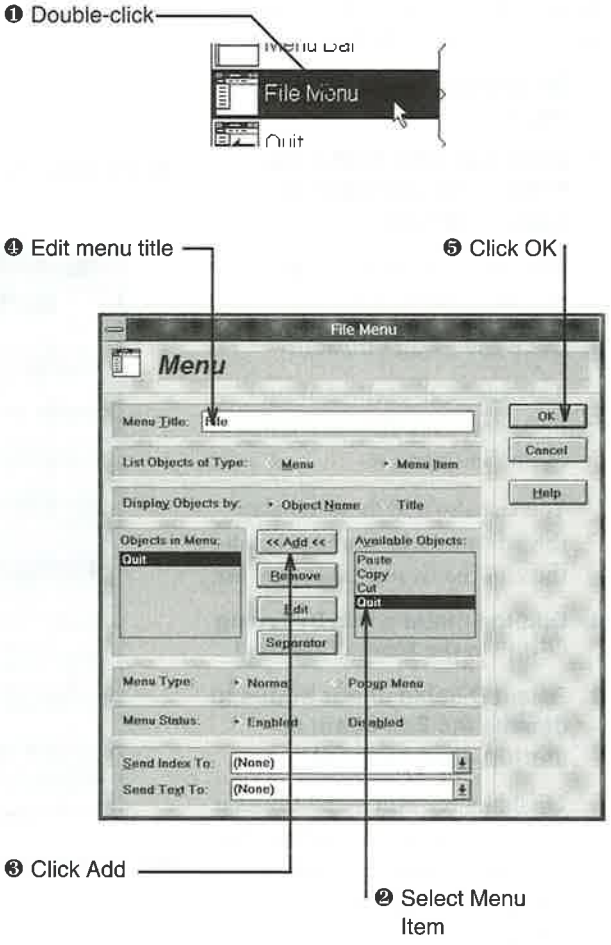
3. **Click the «Add« button.**

   You should see Quit appear in the Objects in Menu list. This indicates that at run time Quit will appear in the File menu.

4. **Edit the menu title by typing "File" in the highlighted Menu Title field.**

   This is the name that appears in the menu bar at run time.

5. **Click OK.**



☛ **Note** You can also add items to the Objects in Menu list simply by double-clicking them in the Available Objects list.

Linking Menu Items to the Edit Menu

As with the File menu, you must link the menu items to the Edit menu.

1. **Double-click the Edit Menu object.**
2. **Click Cut, then Shift-click Paste. This will select Cut, Copy, and Paste.**

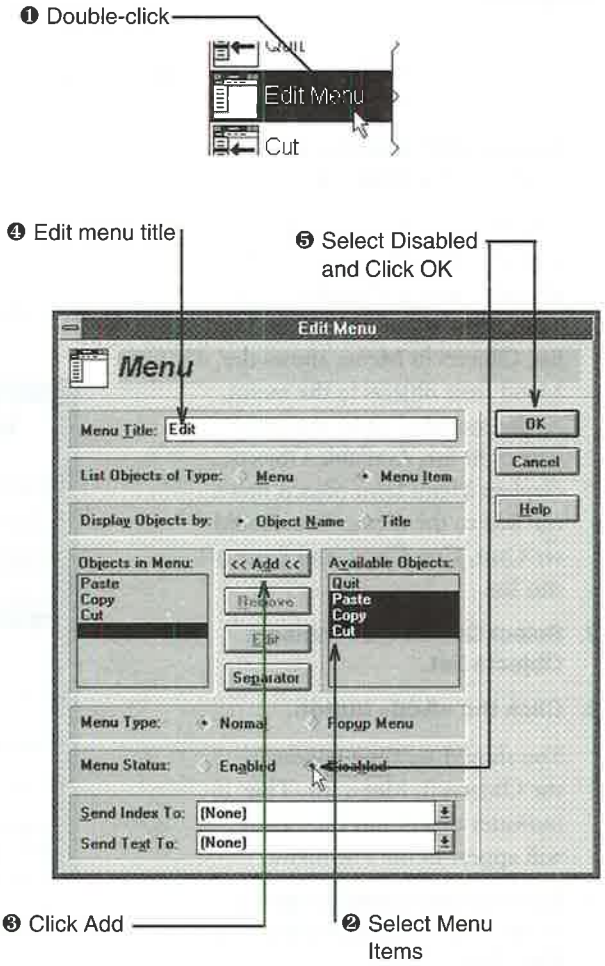
   Also, you can Click on Paste, then drag down to Cut.
3. **Click the «Add« button.**

   This selects all of these menu items and adds them as a group to the Edit menu.

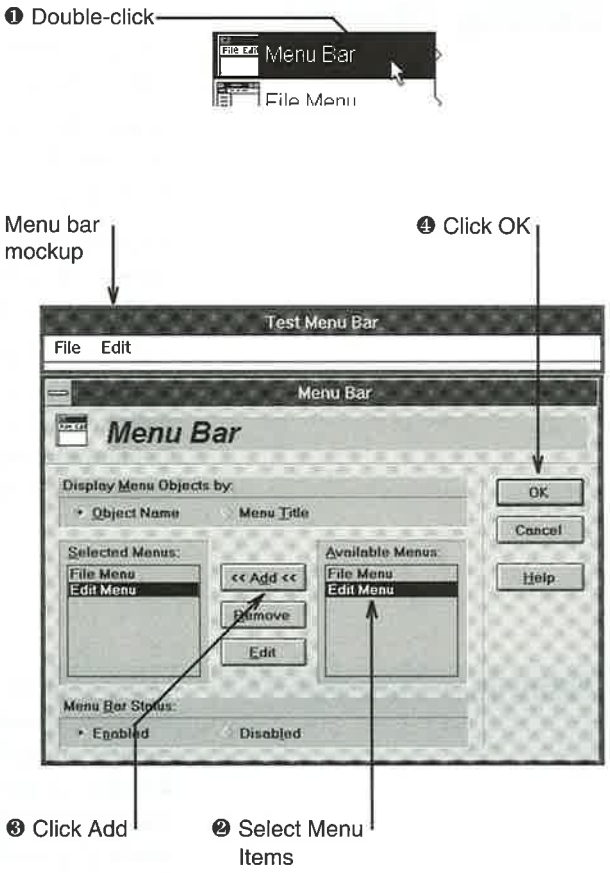
   You can also add these items to the Edit menu by double-clicking them in the Available Objects list.
4. **Edit the menu's title by typing "Edit" in the Menu Title field.**
5. **Select Disable under Status to disable the Edit menu at startup, then click OK.**

   Disabling the Edit menu causes it to be "grayed-out" at startup. You will create a function chain to enable this menu later in this lesson.

You have now configured the File and Edit menus.



Linking Menus to a Menu Bar



Now you need to assign the File and Edit menus to the menu bar.

1. **Double-click the Menu Bar object.**

   The Menu Bar editing dialog displays two lists. The left list, Selected Menus, shows the menus that are part of the menu bar. At the moment, there are none. The right list, Available Menus, displays Menu objects that you can add to the menu bar. You should see the File and Edit Menus in this list.

2. **Shift-select File Menu and Edit Menu in the Available Menus list.**

3. **Click the «Add« button.**

   You should see the File Menu and Edit Menu appear in the left list, indicating that you have put them in the menu bar. You see a mock-up of the menu bar in the Test Menu bar dialog.

4. **Click OK.**

☛ **Note** You can test your menu bar by clicking on the items in the Test Menu Bar window.

You can also add a Menu object to the menu bar by double-clicking the Menu object in the Available Menus list.

Shortcut for Editing Objects

There is a shortcut for building and editing the constituents of your menu bar.

When editing the Menu Bar object, you can double-click any of the Menu objects in the Selected Menus list to bring up its editing dialog.

Likewise, when editing a Menu object, you can double-click its Menu Item objects in the Objects in Menu list to display their editing dialogs.

You've now finished setting up the menu bar. These items have as yet no functionality; they're labels to indicate what you want to happen when they're selected. In following sections, you'll learn how to make the menu items functional.

What is a Function?

Making objects work is the job of functions. Functions are operators that make objects do something specific and useful. Think of objects as nouns and functions as verbs. By combining the two, you create meaningful program statements. Combining them means graphically connecting them. Like objects, functions are represented on screen by icons. When you want an object to do something, you draw a line from it to the appropriate function icon. For example, to make the Quit, Cut, Copy, and Paste menu items work, you must connect them to the Quit, Cut, Copy, and Paste functions. The following sections illustrate how this is done.

Using the Quit Function

Now you're ready to start adding functions to your application. Let's start with the Quit function.

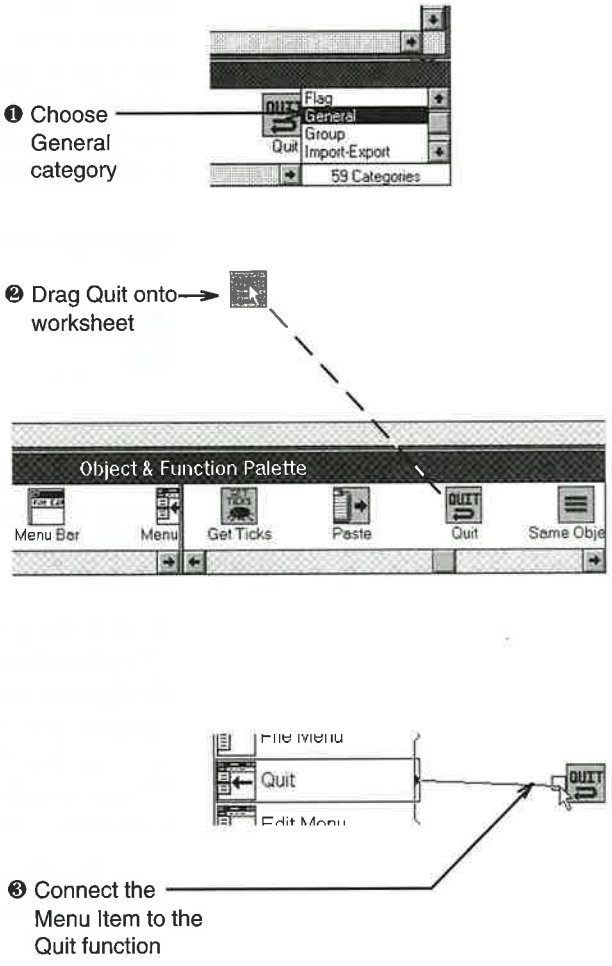
1. **Select General in the category list at the right end of the Objects & Functions Palette.**

   This category list displays the names of all available function types. The total number of types is displayed in the Categories label at the bottom of the list. Selecting a type name in this list displays functions of that type in the function compartment of the Object & Function Palette.

2. **Drag the Quit function out of the Palette onto the worksheet near the Quit Menu Item object.**

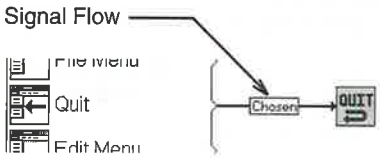
   This creates a copy of the Quit function in your application.

3. **Click just to the right of the Quit object in the object group and drag a line to the Quit function.**



continued...

Note When you click next to the left or right side of a function icon, a small box appears around the area of the click (accept for the Quit function). Similarly, when you click just to the right of an object in the object group, the object is framed by a box. Lines connecting objects and functions can be made from anywhere inside these two boxes. If you click outside the region of the box, no line can be drawn.



The direction in which you draw the line is not important, so you could just as well draw from the right side of the Quit Menu Item object to the left side of the Quit function.

If you've made the connection correctly, you should see a line drawn between the Quit Menu Item object and the Quit function. This line indicates that at run time, the Quit menu item sends a *signal* to call the Quit function, quitting your application. The line is labeled Chosen, to indicate that the signal is issued when the Quit menu item is *chosen*.

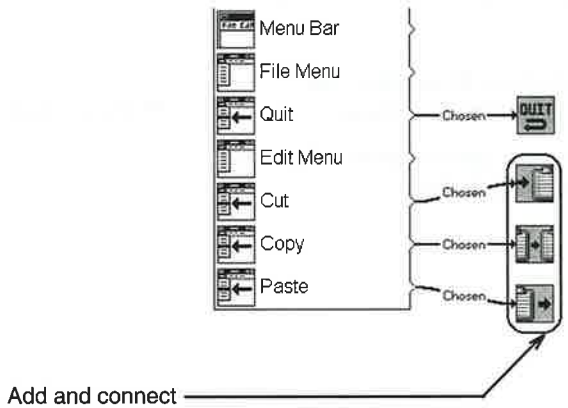
If you click too close to the function icon, you may inadvertently move the icon instead of dragging the connection line. If this happens, just reposition the icon and try again.

Using Cut, Copy, & Paste Functions

To set up the functions for the Edit menu items:

1. **With the General category still selected, locate the Cut, Copy, and Paste functions in the Object & Function Palette. Drag them into the Function worksheet.**
2. **Connect these functions to their respective menu items, just as you did the Quit function.**

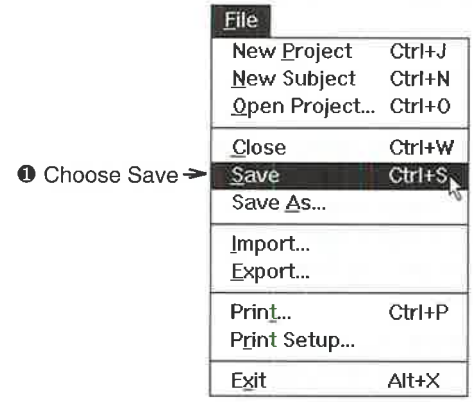
Your menu items are now functional.

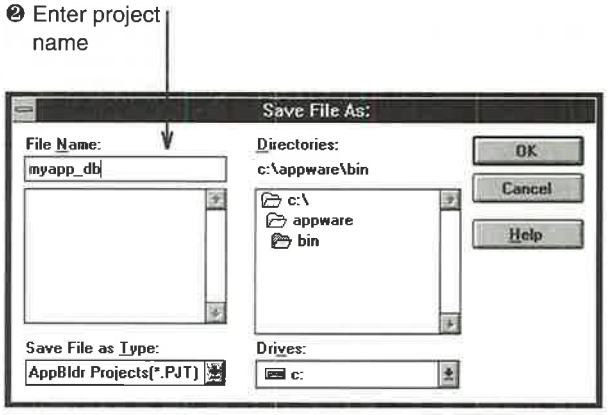


Saving your Project

Take a moment now to save your work:

1. **Choose Save from the AppWare File menu.**
2. **Name your project "MYAPP_DB" and click OK.**

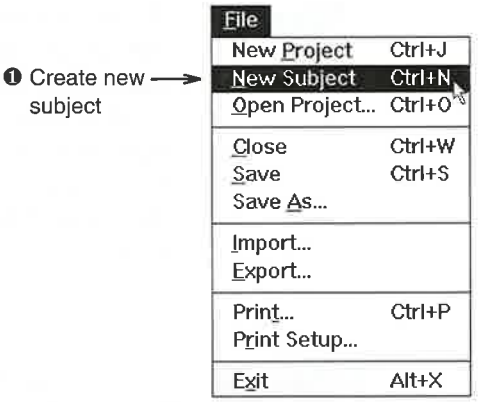




Creating More Subjects

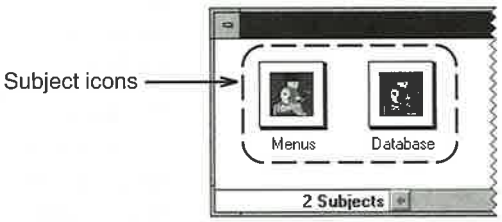
One of the goals of object oriented software is to reuse code sets (objects) from one project to another. Your AppWare project involves no conventional coding, but it does represent work that you would rather not repeat. *Subjects* make reuse possible. In this lesson, you'll learn how to use subjects more effectively.

You'll now create a new subject to hold your Database objects and functions.



1. **Choose New Subject from the File menu or press Ctrl+N. Type "Database" when prompted for the subject name.**

   You now have two subjects in your project: Menus and Database. Their names are *MYAPP_DB.PJT: Menus* and *MYAPP_DB.PJT: Database*, respectively. Since you can have multiple projects open, this naming convention identifies each subject with its parent project, *MYAPP_DB.PJT*.



☛ **Note** You can have as many subjects open as you want. The subjects in your project appear as subject icons in the database project window.

Each subject has its own worksheet. In your new Database subject, you'll include a Window object to contain fields for data entry. You'll also add the Browser and Database objects to give your program its functionality. You'll add several Text objects to serve as fields in your database.

More Objects

The following sections introduce several new objects.

Window

The Window object enables your application to display and control windows that contain other AppWare objects such as Button, Date, List, (pop-up) Menu, Number, Picture, Scale, Text, Table and Time. Your application may have as many windows as you like, each with a unique set of attributes (maximize and minimize buttons, scroll bars, etc.), containing a different group of items.

Database

The Database object is an engine for implementing flat-file and relational, single and multiuser database applications. In connection with other objects provided in the Database ALM set (Browser, Import-Export, Page Layout, and Table), it allows you to build versatile databases involving ASCII file import and export, multi-column tables, and layouts for printing reports, labels, and lists.

The distinguishing feature of the Database object is the ease with which you can integrate it with other objects. Integration works on two levels. First, the Database object can be combined with other objects as part of an integrated solution incorporating wide-ranging functionality. Second, the Database object itself can contain many other objects in its data structure. This means that you can store not only alphanumeric fields, but also pictures, buttons, lists, menus, movies, sounds, animation sequences, and even entire multicolumn tables. The Database's ability to save arbitrary objects as fields of a record enables you to store and retrieve data not conventionally accessible in databases.

Browser Used in conjunction with a Database object, the Browser is a palette of buttons that automatically implement essential database operations, sparing you the bother of having to set up your own buttons and Database function chains. Browser buttons call the same code routines as the corresponding Database object functions. The Browser also lets you call your own function chains in response to any of its signals in order to supplement or replace its built-in operations. While generally not intended for larger, professional applications, the Browser is ideal for quick and easy database setups where a more customized interface isn't needed.

Text The Text object allows your applications to display, edit, and manipulate text.

Adding More Objects

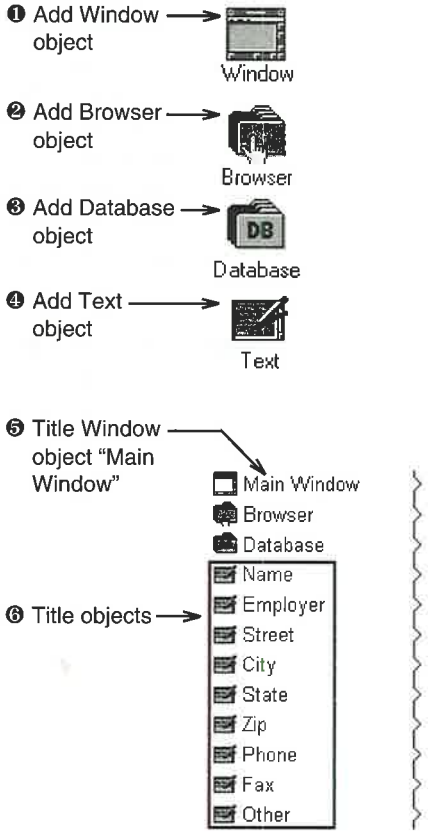
Your database program needs to have a Window object to display editable fields and button controls, Browser and Database objects to give the application database functionality, and Text objects to be used as database fields.

1. **Find the Window object in the Palette and drag it onto the subject worksheet.**
2. **Drag the Browser object into the newly created object group.**
3. **Drag the Database object into the object group.**
4. **Drag the Text object into the object group. Do this eight more times so that there are nine Text objects in the group.**

   You may need to reposition the objects in the object group to match the diagram at right.

   The diagram at right shows objects displayed as Small Icons. You will learn about this feature latter in this manual.

5. **Title the Window object "Main Window."**



6. **Beginning at the top, title the Text objects: "Name," "Employer," "Street," "City," "State", "Zip," "Phone," "Fax," and "Other."**

☛ **Note** The diagram above shows the objects displayed as small icons. Displaying objects as small icons increases the number of objects you can see in an object group. To toggle between large and small object icons, select an object group, then choose Set to Small Icons or Set to Large Icons from the Subject menu. These commands only affect object icons in the selected object group.

Setting Up the Window

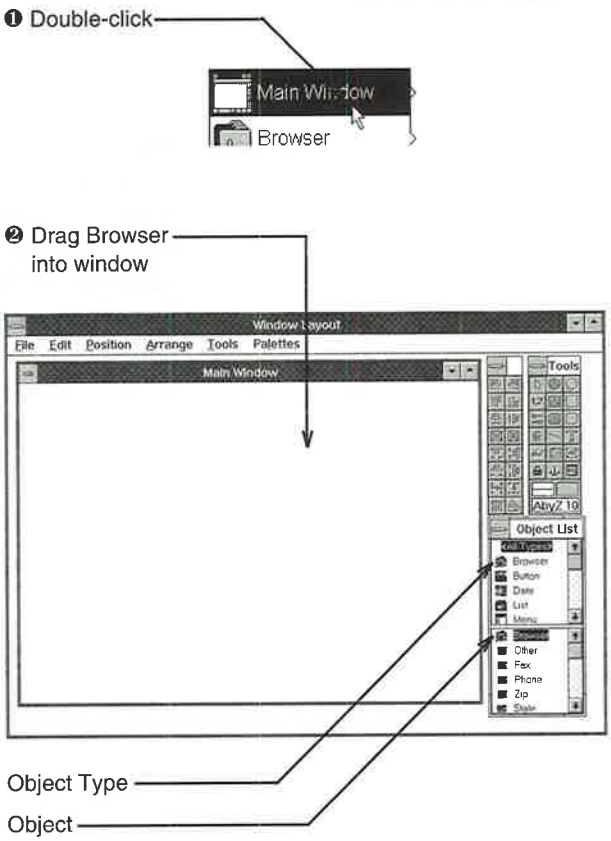
Now that you've given names to the objects in the Database subject, you need to edit them as you did the Menu Bar, Menu, and Menu Item objects. You will begin by editing the Window object to create the user interface for your database application.

1. **Double-click the Main Window in the object group.**

   The blank Window Layout window appears. Here you will design the look of the Rolodex window. You'll do this by adding other objects to the layout, and by positioning and sizing those objects. To create graphical backgrounds for your windows, you'll use the palette of drawing tools located to the right of the window. To move objects from the palette to the window, click and drag them into the window.

2. **Drag the Browser into the main window from the Object List palette.**

   You can also add the Browser to the window by double-clicking it.



☛ **Note** Drag only the *Browser object* (lower list), not the *Browser Object Type* (upper list). Choosing from the upper list creates a new instance of the Browser object in your project.

3. **Now add the nine Text objects to the window just as you did the Browser object.**

   You can select all nine by clicking on Name and then shift-clicking on Other. You can make discontiguous selections one at a time by Ctrl-clicking each item.

4. **Choose Window Attributes from the Edit menu.**

   The Window Attributes dialog appears.

5. **Enter the title "Rolodex" in the Window Title field.**

6. **Click Maximize Button to unselect that attribute.**

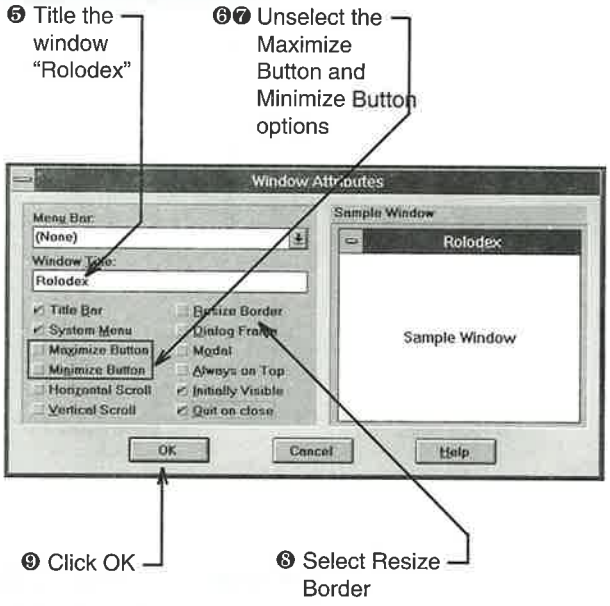
7. **Click Minimize Button to unselect that attribute.**

8. **Click Resize Border to unselect that attribute.**

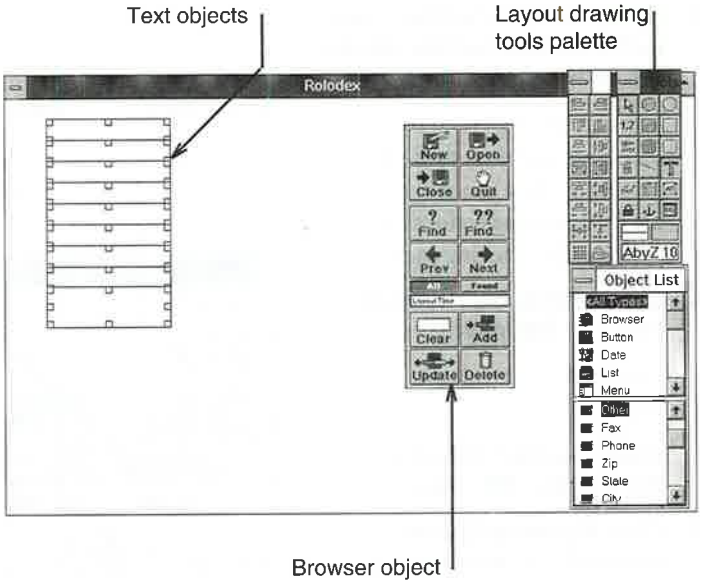
   Steps 6 - 8 configure the window to be that of a "Fixed-Size" window.

9. **Click OK.**

   It is here that you design the look of the Rolodex window. You'll do this by positioning and sizing the objects that make up the window layout. To create graphical backgrounds for your windows, you'll use the palette of drawing tools located on the right side of the window. For now, just move this palette out of the way.



continued...



☛ **Note** You'll use the diagram at the beginning of this chapter as a model for the user interface. The following steps describe how to move objects in the window layout and how to add text and background graphics to match this model.

Layout Grid

The window layout provides a grid to aid in positioning objects in the window. This grid constrains the size and location of drawing shapes and objects to a predefined grid.

To edit the grid settings:

1. **Select the Grid tool in the Tool palette.**

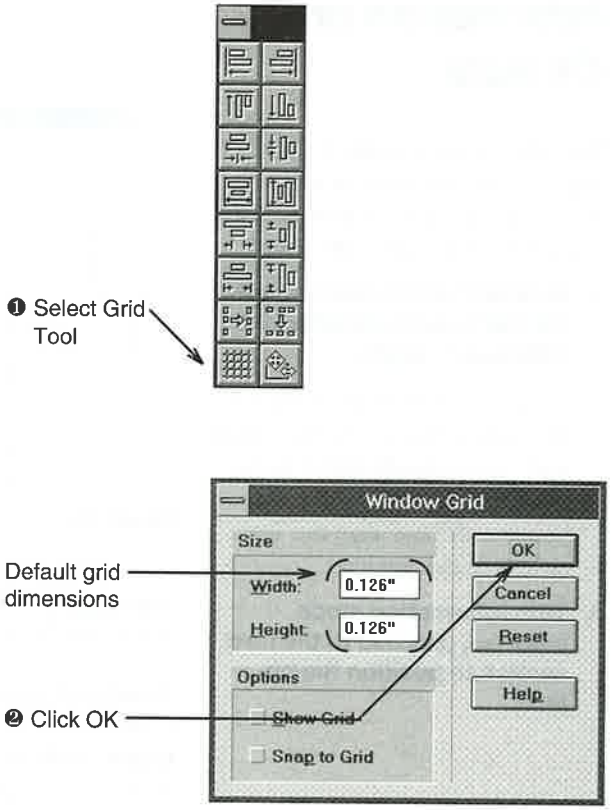
   The Grid dialog appears, allowing you to specify the horizontal and vertical units of the grid. The default unit of measure is inches. The default grid is 0. 000″ x 0.000″.

2. **Set the width and height coordinates to .126″.**

3. **Select the Show Grid and Snap to Grid checkboxes.**

   Show Grid displays the grid in the window layout. Snap to Grid causes objects to align to the grid.

4. **Click OK.**



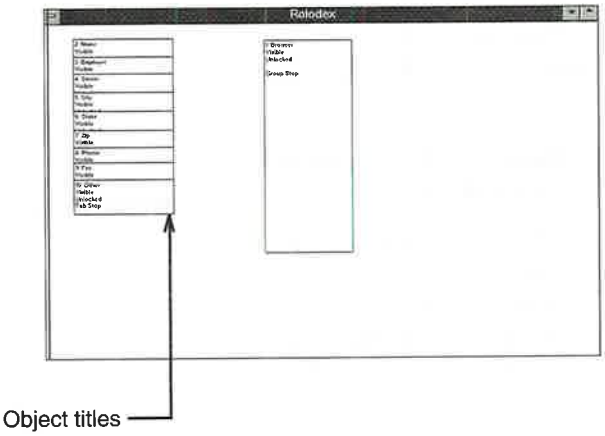
Getting Information on Objects

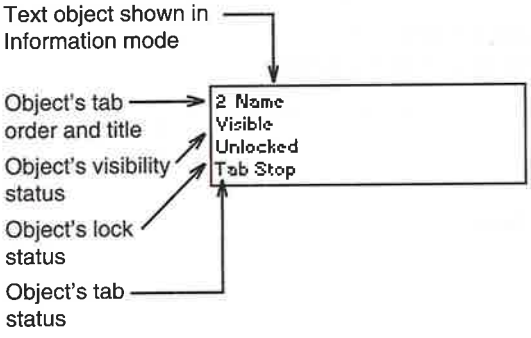
Now that you've enabled the grid, you'll use the Information tool to reveal the titles of the objects so that you know what you're positioning.

1. **Choose Full Information from the Tools menu to select the Information mode.**

   Objects in the window appear as rectangular boxes with their name and object type displayed in the top left corner. These cues assist you to order and align the Text objects in the window.

2. **With information mode selected, proceed to the next section to position the objects.**

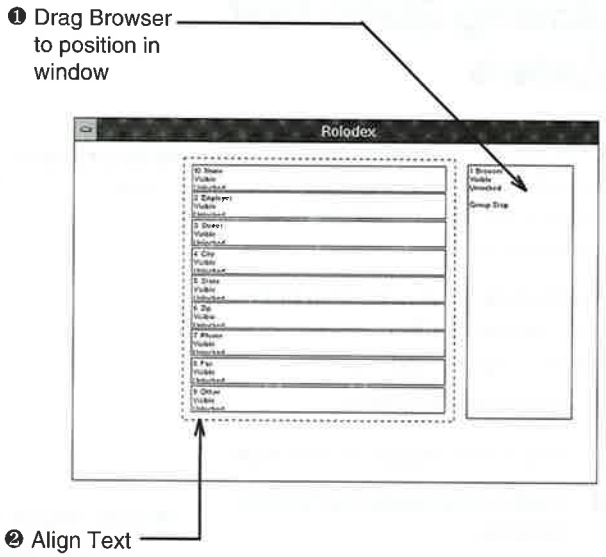


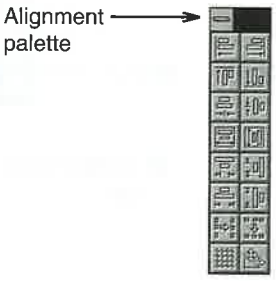


Positioning Objects

1. **Select the Browser object and drag it near the top right of the window.**
2. **Align the Text objects next to the Browser by moving and resizing them.**

   Use the diagram at right as a guide for your application. You may find the tools in the Alignment palette useful in arranging the objects in the window. For more information about the Alignment palette, see the Window chapter in *Essential ALMs*.
3. **After you have aligned the objects in the window, choose Full Information from the Tools menu to deselect the Information mode and proceed to the next section.**





☛ **Note** When positioning the Browser object in the window, make sure that the Browser's rectangle and all of its buttons are entirely visible within the layout window. You may need to resize the layout window to view the Browser object.

If you want to resize this or any other window, click one of its borders and drag it with the mouse.

Adding Static Text Labels

Now that the Text objects have been placed, they need to have labels to indicate the kind of information contained in them.

To label the fields contained in the Text objects you'll create static text labels using the Text tool.

1. **Select the Text tool.**

   The cursor changes to an I-beam.

2. **Click once inside the layout window.**

   The Static Text dialog appears, allowing you (among other things) to enter the text of the label.

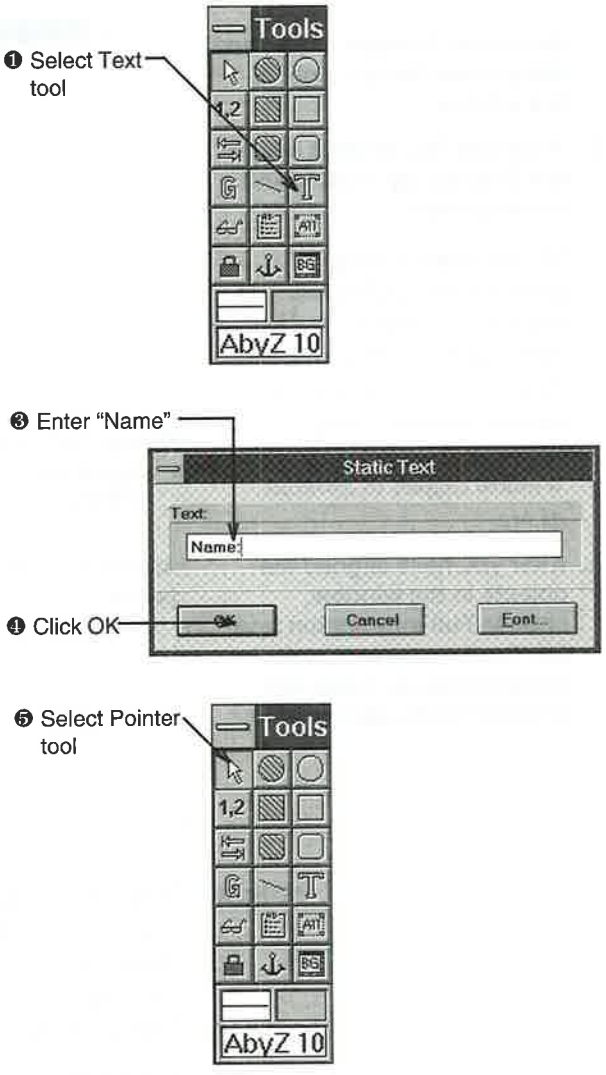
3. **Enter "Name:" in the text box.**

4. **Click OK.**

   A static text box appears with the word "Name:" displayed inside.

5. **Select the Pointer tool or press the right mouse button and align the text box to the left of the first Text object by moving and resizing.**

6. **Repeat steps 1–5 eight more times, naming the labels: "Employer:", "Street:", "City:", "State:", "Zip:", "Phone:", "Fax:", and "Other:".**



Drawing Boxes

Now you'll draw a box around the objects.

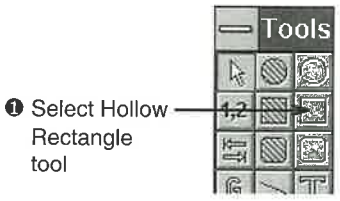
1. **Select the Hollow Rectangle tool.**

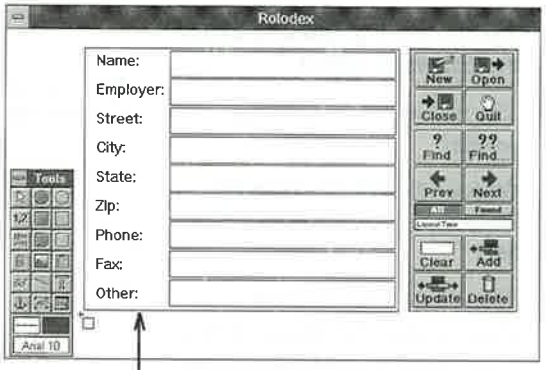
   The cursor changes to the hollow rectangle cursor.

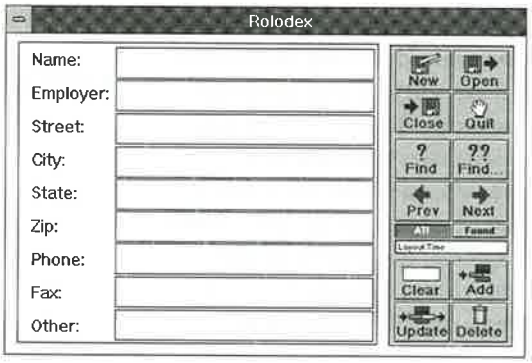
2. **Drag a box around the static text boxes and the Text objects as in the diagram below.**

   You may want to move all the Text objects and resize the window to make the interface look more proportional. Using the Pointer tool, drag a marquee around the objects, the labels, and the box and move them.

A complete color layout is provided in the accompanying Lesson 2 project file located in the Tutorial directory.







Setting Up the Browser

You can configure objects in the layout by double-clicking them. Double-clicking an object in the layout opens the object's editing dialog just as it does in the object group.

1. **Double-click anywhere inside the Browser object.**

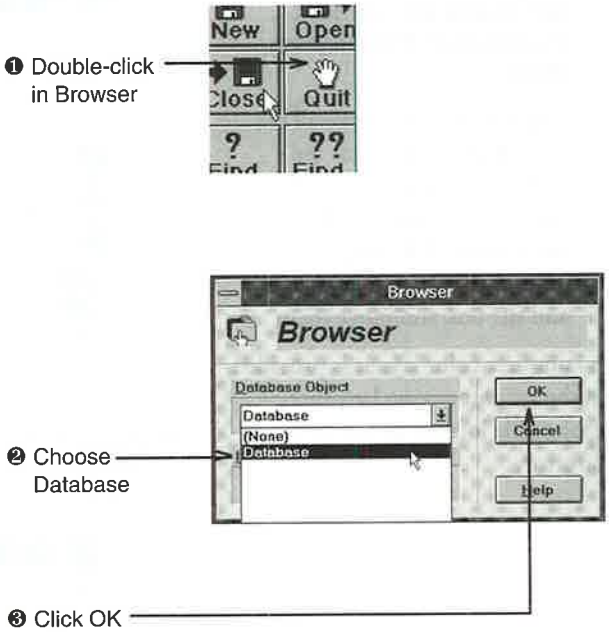
   The Browser editing dialog appears.

2. **Choose Database from the Database Object drop-down list.**

   This tells the Browser which Database object it is to control. In a more complicated database application, you might have several different Database objects to choose from. A given Browser can only control one Database object, and you must explicitly specify it.

   Notice that the drop-down list box named Indexed Field Object is now enabled with the default selection *Active Item*. This selection makes it possible to locate records using whichever field is active, so long as the active field is indexed. It's possible to limit searches to a specific field. For now, leave this selection as it is.

3. **Click OK.**



At this point, you're finished with the window layout. You'll configure the rest of the objects from the subject window.

4. **Double-click the window layout's Control-menu box to return to the subject worksheet.**

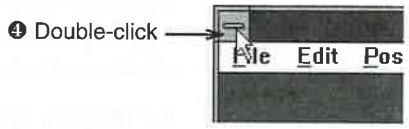
   A dialog appears, allowing you to save any changes made to the Window object.

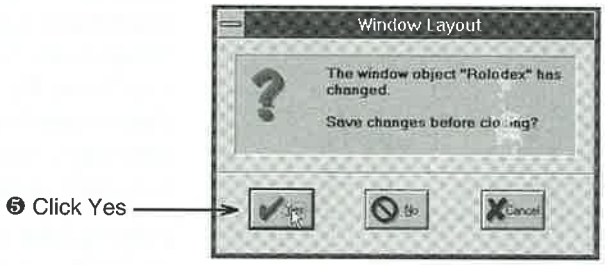
5. **Click Yes to save your changes.**

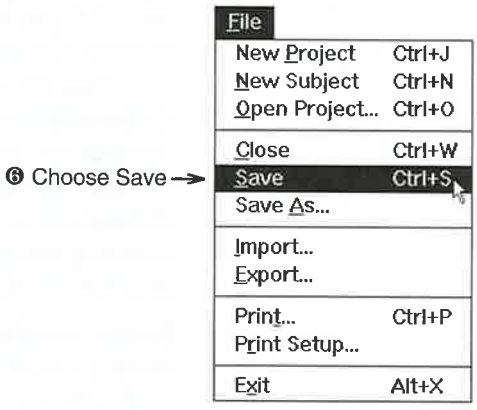
   Take a minute now to save your project.

6. **Choose Save from the File menu.**

   After exiting the window layout, you should also save the project by again choosing Save from the File menu. Saving changes to the window does not update the project file on disk.







The Database Object

The Database object takes care of all of the low-level operations needed to manage database files. Like other objects, it frees you from having to worry about code-level implementation so that you can concentrate your efforts on effective application design.

For the purposes of this application, the only things you need to tell the Database object are what its record structure is to be and which fields are to be indexed. A record is a group of associated fields. In this Rolodex program, the fields that make up a record are the Text objects Name, Employer, Street, City, State, Zip, Phone, Fax, and Other. For every client whose data is stored in the Rolodex, there is at least one record. All of the records taken together in turn make up the Rolodex file.

Indexing fields in a database provides a method of referencing the records in various orders. Each indexed field maintains a reference list, sorted according to data type. For example, a text object sorts in alphabetical order (Ed, Joe, Matt, 1, 11, 12, 2, 21, 22, etc.). Number objects sort in numerical order (1, 2, 12, 13, etc.). Date objects sort in chronological order. What a database file amounts to is the collection of records stored in the order of creation and a set of indexes (one for each indexed field) which reference the data in different orders. Indexing speeds up the process of retrieving records as well as accessing the data in order of different fields.

Because indexed data is presorted, it's easy to find without a lot of searching, so retrieved times are faster. In the interest of performance, therefore, it's a good idea to index commonly referenced fields. The disadvantage of indexing is larger file size.

Setting Up the Database Object

To specify the record structure of the Database object:

1. **Double-click the Database object in the object group.**

   The Database editing dialog appears allowing you to define fields for each record.

2. **In the Available Objects list, click Name and drag down to Other, selecting all text objects.**

3. **Click the «Add« button.**

To apply indexing:

4. **Select the Name object in the Objects in Record list.**

   The first object listed is indexed by default, indicated by the selected Indexed Field checkbox.

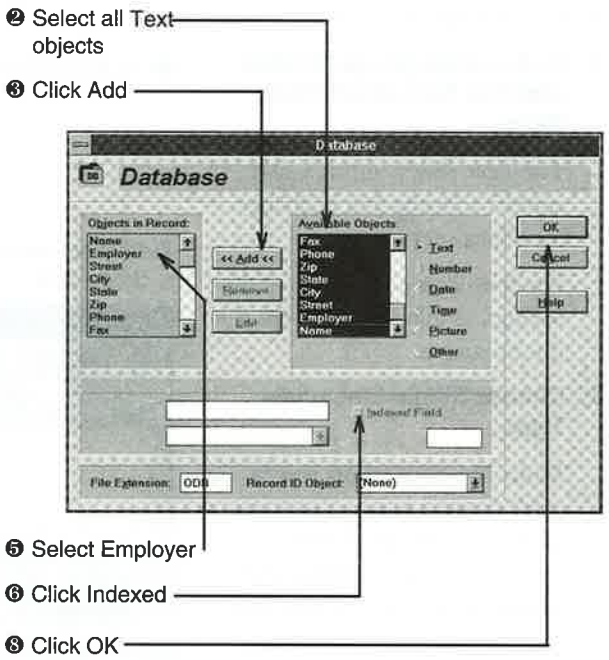
5. **Select the Employer object in the Objects in Record list.**

6. **Click the Indexed checkbox to make the Employer object an indexed field.**

7. **Repeat Steps 1 and 2 for the City, State, and Zip fields.**

   You've now set up the Database object.

8. **Click OK.**



☛ **Note** You should only index those fields that you're likely to use for finding records. Street address, for example, is not a very common criterion for looking up clients in your Rolodex. Name and Employer, on the other hand, are.

Setting Up the Text Objects

Next, you'll edit the Text objects.

1. **In the object group, Double-click the Text object titled Name.**

   The Text object editing dialog appears allowing you to change the object's settings.

   You'll need to modify two of the settings, as they are not needed in this program.

2. **Choose Single-Line under Type.**

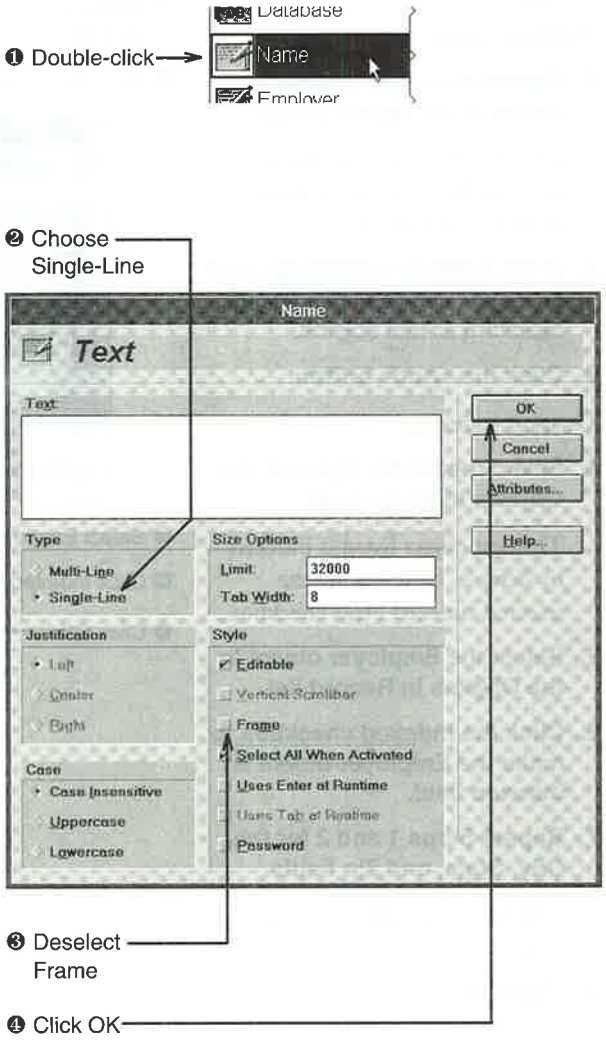
   Multi-Line texts allow you to enter carriage return characters at run time. In the Rolodex, however, the Text objects each contain only a brief, single line of text.

3. **Deselect Frame.**

   The Frame option displays a thin-line frame around the Text object. In the layout style we've chosen, a separate frame for each object is unnecessary.

4. **Click OK.**

5. **Repeat Steps 1-4 for the remaining Text objects.**



Sharing & Aliasing Objects

The topic addressed in the following section is how subjects share each other's objects. For objects to reference each other, they must first be "visible" to one another. Recall, for example, that in order to get the File menu in the menu bar, you had to double-click the Menu Bar object, select the File Menu object, and add it to the list of objects owned by the Menu Bar. However, objects can only see each other if they reside in the same subject. The purpose of *object aliasing* is to make objects in one subject accessible or "visible" in another. An *object alias* is a virtual object resident in one subject that represents a real object or *original* resident someplace else. You use the alias as if it were the original. In this section, you'll create an alias of the Menu Bar and put it in the Database subject so that you can associate the Menu Bar with the Main Window. You'll also create an alias of the Edit Menu in the Menus subject and put it in the Database subject. You'll connect functions to this alias in one of the following sections.

Adding a Menu Bar to the Window

The Menu Bar object must be associated with a window in order to be displayed. Since the Menu Bar and the Main Window reside in different subjects, you must share the Menu Bar and alias it in the Database subject before it can be added to the Main Window.

1. Open the Menus subject.

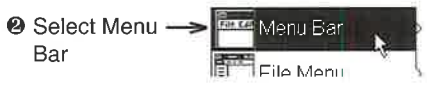
You can get to the Menus subject in any of the following three ways.

a) Choose MYAPP_DB:Menus from the AppWare Windows menu.

b) Double-click the arrow to the left of MYAPP_DB in the Navigator window, then Double-click Menus. The Navigator is displayed when you choose Navigator from the Windows menu.

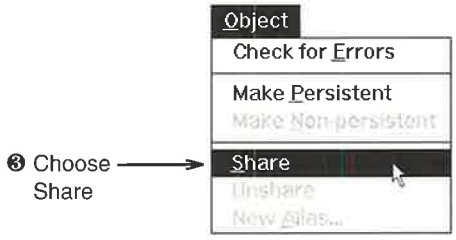
c) Double-click the Menus subject tile in the project window.

2. Select the Menu Bar object in the object group.



3. Choose Share from the Object menu.

Once an object has been shared, its title is bold face in the subject's object group. The shared object is now available for aliasing.



4. Open the Database subject.

5. **Choose New Alias from the Object menu.**

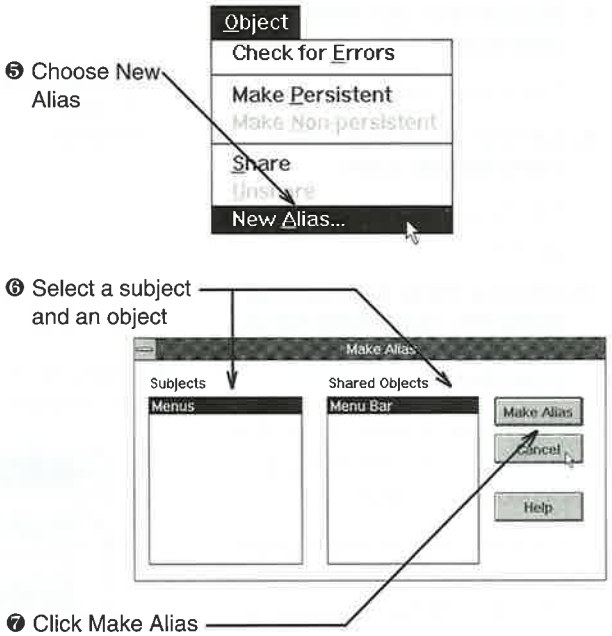
   The Make Aliase dialog appears allowing you to select an object to alias in the current subject.

6. **Select Menus from the Subjects list, then select Menu Bar from the Objects list.**

7. **Click Make Alias.**

   You should see the Menu Bar appear in the object group. The title is italicized to indicate that the item is an alias.

   Now that the Menu Bar is shared and aliased, you need to add it to the Main Window.



☛ **Note** The Object alias appears in the selected group. If more than one group is selected, or if no group is selected, a new group is created for the alias.

To move an object from one group to another, select the object and drag it into the new group. As the cursor enters the destination group, you'll see a hatched line indicating the insertion point. Move the cursor to the desired location and release the mouse button.

8. **Double-click the Main Window object.**

   The layout window appears.

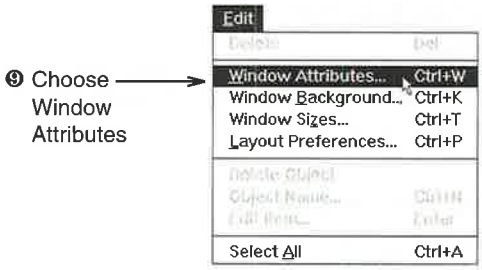
9. **Choose Window Attributes from the Edit menu.**

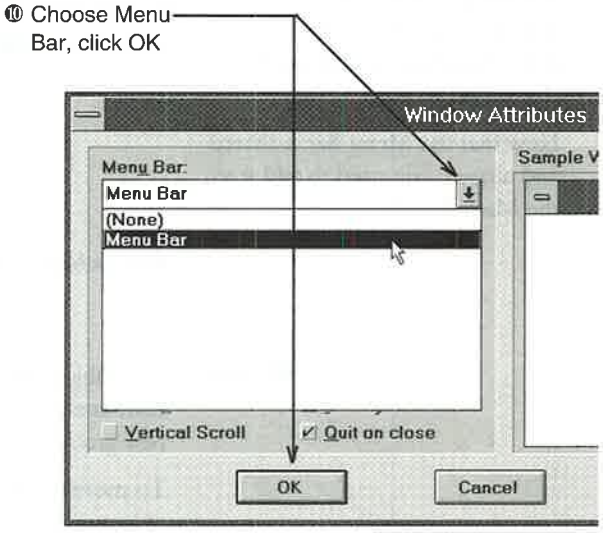
   The Window Attributes dialog appears.

10. **Choose Menu Bar from the Menu Bar drop-down list at the top left of the dialog, then click OK.**

You may need to resize the window in the Window Layout dialog to accommodate the Menu Bar.

To resize the window you will need to select the Resize Border option in the Window Attributes dialog. See page 2-31.



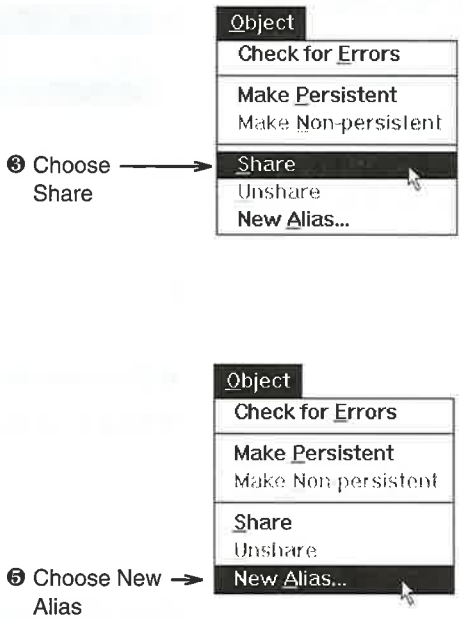


Aliasing the Edit Menu

1. **Open the Menus subject.**
2. **Select the Edit Menu object in the object group.**
3. **Choose Share from the Object menu.**
4. **Open the Database subject.**
5. **Select New Alias from the Object menu.**

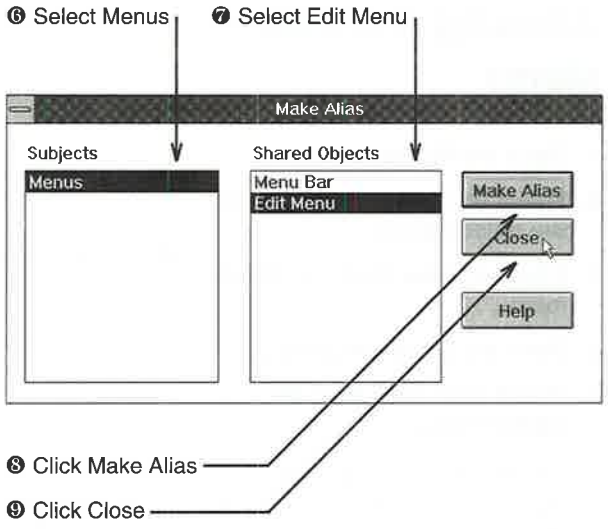
   The Aliasing dialog appears, allowing you to select an object in the current subject.

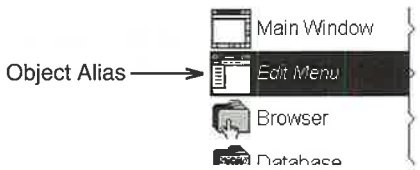
continued . . .



6. **Select Menus from the Subjects list.**
7. **Select Edit Menu from the Objects list.**
8. **Click Make Alias.**
9. **Click the Close button.**

   You should now see *Edit Menu* in the object group of the Database subject.





☛ **Note** It's possible to share and alias multiple objects at the same time. When sharing, hold down the Shift or Ctrl keys. Shift allows you to select multiple, contiguous objects. Ctrl lets you select discontiguous objects. Either key lets you select objects in separate groups.

To alias multiple objects, drag the mouse over the items in the Shared Objects list. Shift – and Ctrl – selection also operate in this list.

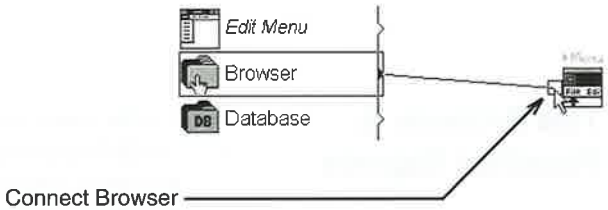
Adding Function Chains

In this section, you'll connect functions to the Browser object to enable and disable the Edit Menu. To eliminate confusion and safeguard against inadvertent loss of data, it's a good idea to disable menu items and buttons when they're not applicable. In the Rolodex, for example, there is little point in using the Edit menu when no file is open. On the other hand, as long as a file is open and its records are editable, the Edit menu should be active. This process may sound complicated, but all it requires is two function chains.

You'll first create the function chain to enable the menu.

1. **Select the Menu category in the Object & Function Palette.**
2. **Find the Enable Menu function and drag it onto the worksheet near the Browser object.**
3. **Click just to the right of the Browser object and drag a line to the Enable Menu function.**

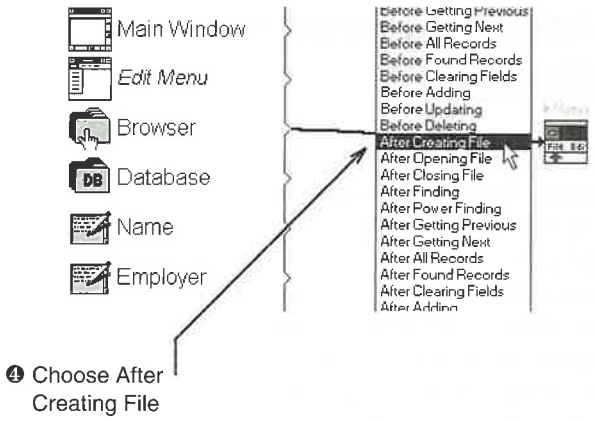
   A signal flow appears between the Browser and the Enable Menu function. The signal is labeled Clear Fields. You'll change the signal to enable the menu after the Browser creates a file.



Connect Browser object to the Enable Menu function

continued...

4. **Choose the After Creating File signal by holding the left mouse button on the signal and scrolling through the available signals.**



The Browser & Function Signals

The Browser is a control panel of buttons that automatically handle basic database operations such as creating, opening, and closing files, adding, updating, and deleting records, etc. At run time, when you click a Browser button, it performs an operation, without your having had to set up any function chains. There are times, however, when you want to make function calls of your own before or after the Browser operation has executed. You may even want to replace the Browser's built-in functionality with a customized function chain you've designed. In order to do this, you need some sort of indication from the Browser that it has just performed an operation or that it is about to do so. That's what the Browser's signals do. Each Browser button has two associated signals, one issued before the operation named by the button is performed, the other after. By attaching function chains to these signals, you can call your own function chains either before or after a Browser operation. In the present case, you want to call the Enable Menu function after the New button has been pressed and its associated operation (the creation of the file) has been carried out.

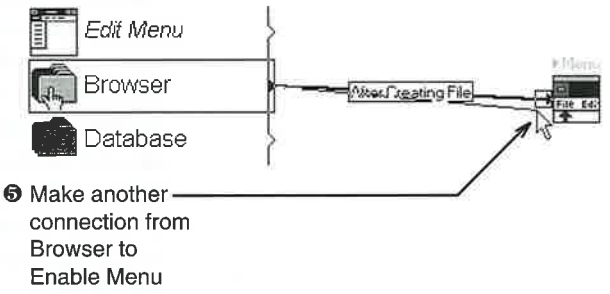
You'll now create another signal flow between the Browser object and the Enable Menu function to enable the menu after the Browser opens a file.

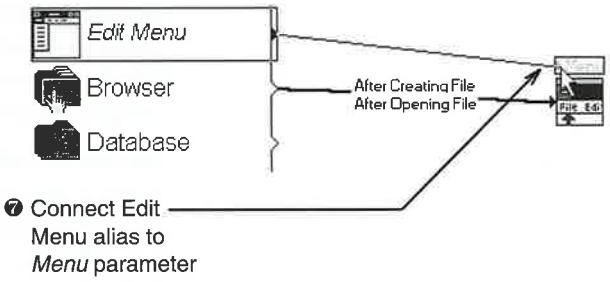
5. **Click just to the right of the Browser object and draw a second signal flow to the Enable Menu function.**
6. **Choose After Opening File from the signal flow pop-up menu.**

   You now have two separate signal flows from the Browser object to the Enable Menu function. One flow is used to enable the menu after the Browser creates a file, and the other flow is to enable the menu after opening a file. However, the Enable Menu function does not know which menu to enable. You need to connect a parameter to the Enable Menu function, instructing it to enable the Edit Menu.

7. **Drag a line from the aliased Edit Menu object to the Menu label above the Enable Menu function.**

   You can also make parameter connections by clicking on the label with the right mouse button and selecting Edit Menu from the pop-up list that appears.





What is a Parameter?

Functions, by definition, operate on objects. The object or objects to which a function does something are its parameters. Parameters are of two types: inputs and outputs. Only input parameters concern us here.

The labels above a function icon are inputs to which you pass parameters. You pass an object as an input parameter to a function by drawing a line between the object and one of the labels above the function icon. To draw this line, click either just to the right of the object or in the label, and drag the mouse. The parameter line you draw from an object to an input label or from an input label to an object is visible as long as you hold the mouse button down. The direction in which you draw the line is unimportant.

Once the tip of the line is within a few pixels of the destination object or label, a box appears around the destination, indicating that a connection can be made. If no box appears, the parameter connection you're trying to make is invalid.

The box around the object or the function label helps to ensure that you don't inadvertently make a connection to the wrong kind of object. When you release the mouse button, the parameter line and box disappear. At the same time, the text of the label changes to reflect the title of the connected object, and the label text turns from gray to black.

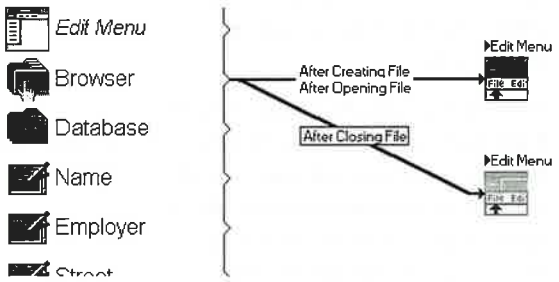
Another way to link a parameter to an object is to click the parameter label with the right mouse button and choose the object from the pop-up list. The objects displayed in this list are filtered by type, so only permitted connections can be made.

To delete a parameter connection, click the function label and press the Delete key.

Disabling the Edit Menu

Another function chain needs to be connected to the Browser object in order to disable the Edit menu after the Rolodex program closes a file.

1. **Find the Disable Menu function and drag it onto the worksheet below the Enable Menu function.**
2. **Draw a signal flow from the Browser object to the Disable Menu function.**
3. **Choose After Closing File from the pop-up menu in the signal flow.**
4. **Draw a parameter line from the aliased Edit Menu object to the top of the Disable Menu function.**



Clearing Fields

In many databases, it's conventional to clear fields after operations such as adding and deleting records or closing a file.

You can manually clear fields using the Clear button. Unlike other Browser buttons, Clear does not automatically perform the operation indicated by its name. It does, however, issue a signal that you can use to call your own function chain for clearing fields. To set up this chain:

Browser Clear button →

1. **Select the Text function category in the Object & Function Palette.**
2. **Drag the Clear Text function into the worksheet and place it opposite Browser.**
3. **Draw a signal flow between Clear Text and Browser.**

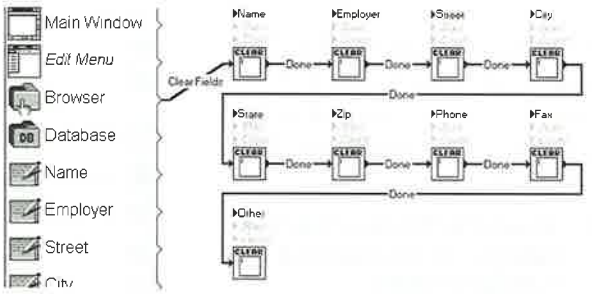
   The flow is labeled Clear Fields, indicating that the signal is issued when the Clear Fields button is pressed.
4. **Draw a parameter line from the Text input above the function to the Text object called Name.**

   This parameter specifies the Text object you want to clear.

Since a given Clear Text function can only clear one field, you need to set up Clear Text functions for each of the remaining fields in the Database (Employer...Other).

5. **Drag eight more Clear Text Functions into the worksheet, placing them as shown in the following diagram.**
6. **Draw a signal flow from the first Clear Text function to the second, from the second to the third, and so on until all the Clear Text functions are chained together.**
7. **Draw a parameter line from the Text input above the second Clear Text function to the Text object called Employer.**
8. **Repeat Step 7 for the rest of the Clear Text functions, connecting their Text inputs to the rest of the Text objects.**

When you're done, your function chain should look like the diagram at right.



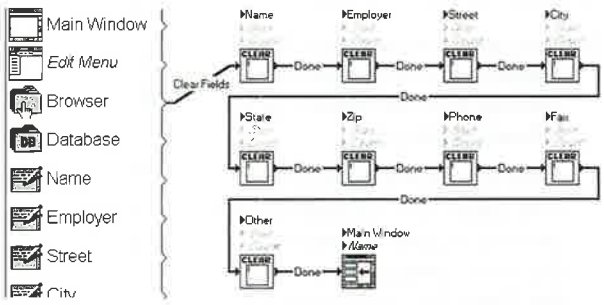
To keep the above diagram uncluttered, the functions shown on page 2-50 are not shown here.

☛ **Note** There is a simpler but slightly more technical method for clearing fields involving the use of the Bundle object described in the *Essentials ALMs*.

The Browser issues the Clear Fields signal when you press the Clear button and also after certain other events such as closing a file or deleting a record. To clear fields after events such as adding a new record, you'll need to draw another flow between the function chain for clearing fields and the Browser, then select the After Adding signal.

Activating Text Field

After clearing fields, you'll often want to add a new record, so it's convenient to have the application place the cursor in the Name field. To do this:



1. **Select the Window function category in the Object & Function Palette.**
2. **Drag the Activate Item function onto the worksheet and place it after the last Clear Text function.**

   The Activate Item function positions the cursor in a specified field (object) in a specified Window object.
3. **Draw a signal flow between Activate Item and the last Clear Text function.**
4. **Draw a parameter line from the Window input label above the Activate Item function to Main Window.**
5. **Draw another parameter line from the Item input label above Activate Item to Name.**

This completes your design work on the basic database application of Lesson 1. Take a minute to save your project. The following sections show you how to create an executable file and run it independently of the AppWare environment.

Compiling & Running Your Application

You're ready now to compile and run your program.

1. **Choose Create Application from the Project menu.**

   In this dialog, you can give your application a name and choose the directory in which to save it.

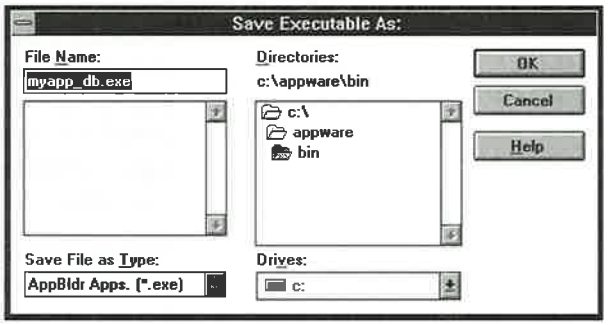
2. **Name your application, choose a directory to save it to, and click OK.**

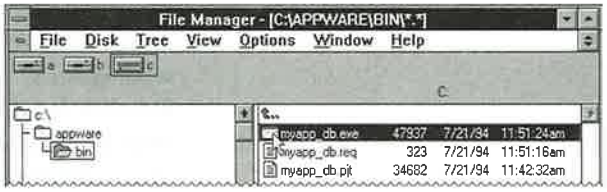
   After AppWare has finished building your application:

3. **Exit from AppWare.**

4. **Find your application in the File Manager and Double-click its icon.**

If the application doesn't work quite right, reopen the project and review your work. Make sure functions and their parameters are set up as outlined in this lesson. Recompile and try it again.





☛ **Note** For more information on how to run the Browser Buttons in your Rolodex application, see the Browser description in *Essential ALMs.*

Delivering Compiled Applications

The resources that make your compiled applications work are not actually included inside the executable file. Rather, all of the program resources are located in DLL files installed in a directory called \APPWARE\BIN. As new versions of the objects are released, replacing these files automatically upgrades applications you've already compiled. For example, if Novell releases a new version of the Database object that offers improved speed in finding records, replacing the Database DLL enables compiled database applications to benefit from the higher performance object without having to be recompiled.

After you have built and compiled a new application, you may wish to move or copy it from time to time. Since your compiled application relies on the object DLL files installed on your computer, you will need to copy both the application and these files to the destination computer in order to run the program.

Upon installation, a directory called \APPWARE is created in the root directory. Inside the AppWare directory, two subdirectories, \BIN and \CONFIG, are created. The first is for AppWare and the ALM (Application Loadable Module) files containing the code that makes objects operate at run time. These files have a .DLL extension. There is generally one DLL file per type of object (ALM). The second directory contains CFG or configuration files that hold the object and function icons, parameter information, and other settings that are used only by the AppWare environment at design time. These files have a .CFG extension and are not accessed by the built applications at run time.

LESSON 2
IMPORT & EXPORT

Topics

- Creating a Subject for Import & Export
- Introduction to the Import-Export Object
- Aliasing Application Fields for Import-Export
- Editing the Import-Export Object
- Defining Field Objects for Import & Export
- Adding Import & Export Menu Items
- Import & Export Process
- Using Import–Export Functions
- Configuring Import & Export
- Starting Import-Export
- Stopping Import-Export
- Creating an Import Chain
- Creating an Export Chain
- Export Process & Loops
- Exclusive Mode
- Turing Off Multiuser Access
- Compiling Your Application
- Running Your Application
- Copying Compiled Applications
- Titling Field Objects (Optional)
- Configuring Import (Optional)
- Specifying the File Format
- Matching Objects to Fields
- Importing Field Names
- Displaying Status Dialog at Run Time
- Configuring Export (Optional)

In this lesson, you'll implement menu item commands for importing data into and exporting it out of your Rolodex. While it's hoped that you'll learn something about import and export, the primary objective of this lesson is to provide additional practice in the use of object aliasing and function parameters. When you finish this, your Rolodex application will be complete, and you'll be ready to start your own AppWare programming.

To begin Lesson 2, open the project you completed in the previous lesson or the Lesson 1 project in the Tutorial directory installed with AppWare.

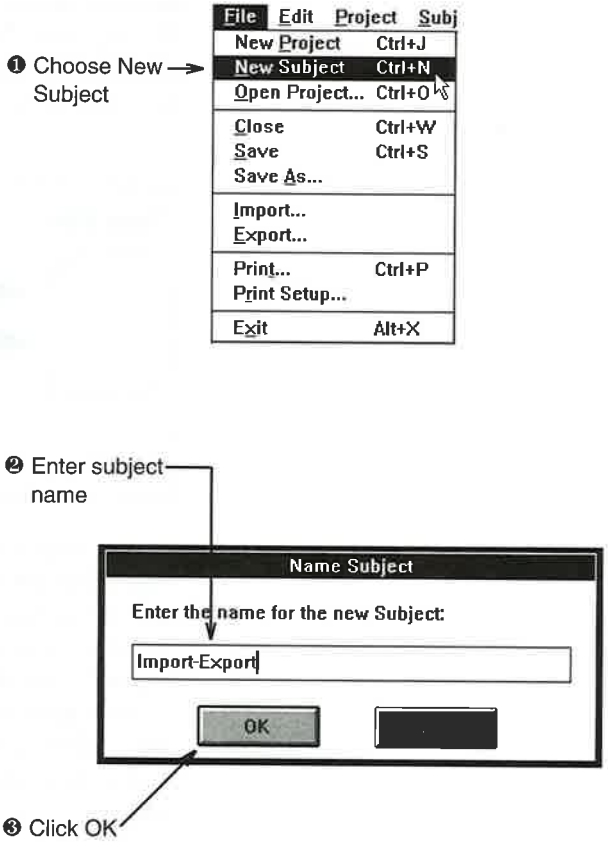
☛ **Note** This lesson concludes with three optional sections that are provided by way of additional explanation. You can skip these sections and still successfully complete the tutorial. You may want to return to them once you start using import and export at run time.

Creating a Subject for Import & Export

The function chains required for importing and exporting data are often much the same from one application to another. In other words, they're reusable. It's a good idea to put these functions and their associated objects in a subject of their own. You'll be able to use this subject again when you start building your own database projects.

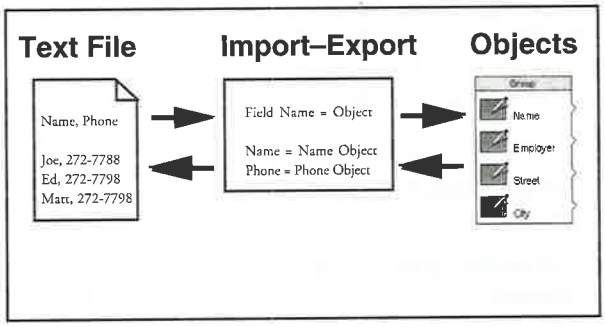
To create the import-export subject:

1. **Choose New Subject from the File menu.**
2. **Enter the name "Import-Export."**
3. **Click OK.**



Introduction to the Import-Export Object

The Import-Export object provides high-level services for data import and export much as the Database object does for data storage, indexing, and retrieval. The import half of the object scans an ASCII text file and copies it line by line to objects in your application. The export half copies data from objects in your application to an ASCII text file. Any object that has some sort of textual representation (Button, Date, Menus, Number, Text, Time) can participate in the import-export process.



A single Import-Export object can both import and export data, although it can't do both simultaneously. This means that in most applications, including the Rolodex you're currently building, you need only one Import-Export object. You'll add this object to your new Import-Export subject in just a minute. So that this object will have fields to reference, however you first need to share and alias the text fields in the Database subject.

Each line in the text file is divided into a certain number of fields. In this illustration, lines in the text file contain two fields, Name and Phone.

Aliasing Application Fields for Import-Export

The fields into which you'll import and from which you'll export data are the nine database fields you created in Lesson 1. Since they're in another subject, you must alias them in the Import-Export subject before you can use them with the Import-Export object.

1. **Open the Database subject.**

   You can get to the Database subject in any of the following three ways.

   a) Choose MYAPP_DB:Database from the AppWare Windows menu.

   b) Double-click the arrow to the left of MYAPP_DB in the Navigator window, then double-click Database. The Navigator is displayed when you choose Navigator from the Windows menu.

   c) Double-click the Database subject tile in the Project window.

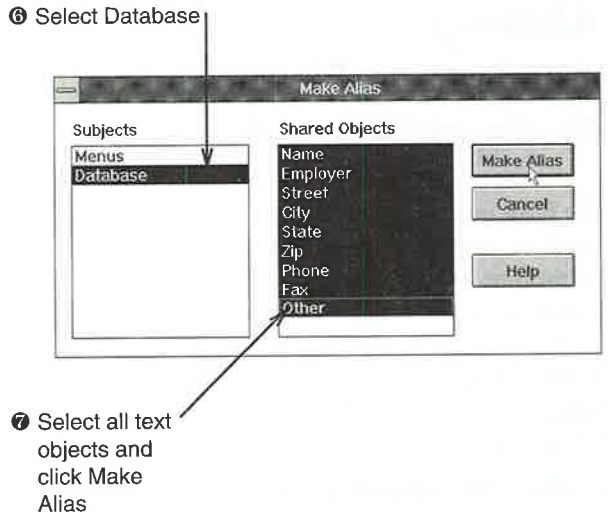
2. **In the object group, select Name, then Shift-select Other.**

   All nine fields should now be selected.

continued...

3. **Choose Share from the Object menu in the AppWare menu bar.**
4. **Return to the Import-Export subject.**
5. **Choose New Alias from the Object menu to display the Aliasing dialog.**
6. **Select the Database item in the Subjects list.**

   When you select a subject in the Subjects list, the shared objects in that subject are displayed in the Objects list.
7. **Drag down through the Shared Objects list to select the nine text fields and click Make Alias.**



☛ **Note** You can also make aliases by double-clicking the selection(s) in the Objects list.

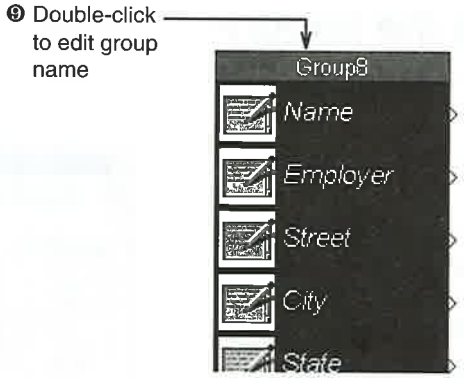
A new object group is created in the Import-Export subject containing the aliases of the text fields. The aliases are now accessible to objects and functions in this subject.

The Cancel button changes to Close.

8. **Click Close.**
9. **Double-click the title bar of the object group.**

   The Group Name dialog appears.

10. **Type "Import-Export" in the edit field and click OK.**



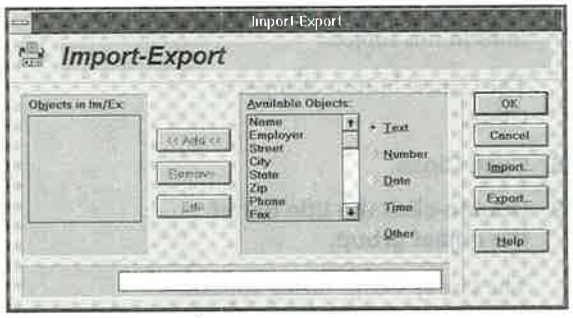
Editing the Import-Export Object

You're now ready to edit the Import-Export object itself.

1. **Drag the Import-Export object icon from the Object & Function Palette into the Import-Export object group.**
2. **Double-click the Import-Export object in the object group.**

The Import-Export editing dialog appears.

This dialog lets you determine which objects in your application can participate in the import-export process and what their run time names will be.



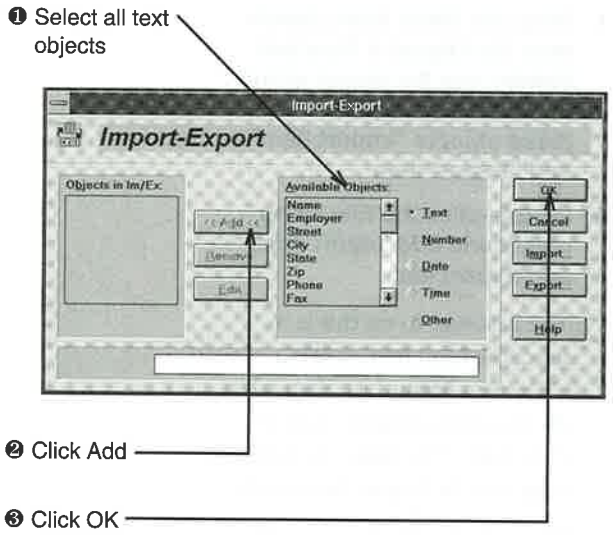
Import-Export dialog

Defining Field Objects for Import & Export

Before an object can participate in the Import-Export process, you must add it to the Import-Export Object:

1. **Select the nine objects Text objects you just aliased (Name...Other) from the right list of the editing dialog.**
2. **Click the Add button.**

   The objects are added to the Import-Export object and their titles appears in the left list. Objects making up this list may receive data during import and provide data during export.
3. **Click OK.**



☛ **Note** To remove one or more objects from the Import-Export Objects list, select the object(s) you want to remove and click the Remove button.

Adding Import & Export Menu Items

You now need to add Import and Export menu items to the File menu.

1. **Drag two Menu Item objects from the Object & Function Palette into the object group of the Menus subject. Title these objects "Import Item" and "Export Item."**
2. **Double-click the File Menu object and add Import Item and Export Item.**

   The best way to do this is to select the Quit item in the Objects in Menu List, then select the two menu items to add and click Add. This places the Import Item and the Export Item in the correct order (before the Quit Item) in the File Menu. Otherwise, you would need to delete the Quit Item, add the two new menu items, then add the Quit Item again.
3. **Open the Import Item and Export Item editing dialogs, enter the names "Import" and "Export" in the Startup Text field, and click OK.**
4. **Share the Import Item and the Export Item objects, then alias them in the Import-Export subject.**

Import & Export Process

During import, the Text object Name receives data from the name field in the file you're importing; the Employer object receives data from the employer field, and so on. After a person's information has been read in from the file, you call the Database function Add Record to create a record in the database consisting of these nine fields. A new line of information is then read in, consisting again of name, employer, street, etc., and another database record is created. This process continues until there are no more lines of information to import.

During export, this process works in reverse. You use a Database function to retrieve a person's record, consisting of name, employer, etc., and the fields of this record are then exported to a text file. The exported record is a single line of text comprising fields delimited by commas or tabs. Once this record has been exported, another is retrieved from the database and exported, and so on, until there are no more records to export.

Using Import-Export Functions

This section introduces you to the Import-Export functions that you'll be using shortly to program the import and export processes. Importing and exporting typically involve the following steps.

Specifying the File to Import

Use Choose I/E File to choose the file to import. At run time, if no pathname is supplied, this function displays the Choose Import File dialog, allowing the user to choose the file.

Specifying the File to Export

Use Create Export File to create a new export file. At run time, if no path name is passed to it, this function presents the user with the standard Save dialog and optionally returns the file name entered by the user. Create Export File stores the file name in the export portion of the Import-Export object so that it knows where to export data. If you do not want the file dialog to be displayed, you, the builder of the application, can supply the complete pathname to the export file, in which case a file is created and located according to the path name.

A proper path name is of the form:

Drive:\Directory\...\Directory\Filename.ext

To append exported data to an existing text file, use Choose I/E File. At run time, this function displays the Choose Import File dialog,, allowing the user to choose the file to which exported data is to be appended. You can optionally supply a path name, in which case the dialog is not displayed.

Configuring Import & Export

Use Configure Import and Configure Export if you wish to allow the user to set importing and exporting capabilities for your application at run time.

The Import/Export Object allows you to predefine the Import and Export configuration. The configuration dialogs that appear at run time are similar to the configuration dialogs displayed by the Import/Export object. However, the option for displaying the status dialog is not available to the user.

Starting Import-Export

Use Start Import-Export to open the text file and set up the import/export process. This function should be called to start both import and export.

Next, if you're importing records, call Import Record to import a line from the text file. Generally speaking, you'll call Import Record within a loop since you'll likely want to import more than one line of data at a time. Loops can be set up either by physically creating a circle of functions on the screen or by using the Loop object (see *Essentials ALMs*).

If you're exporting data, call Export Record to export a line to the text file. As with Import Record, this function is usually called within a loop, since exporting typically involves getting multiple records in succession.

The import/export direction is established the first time you use Import Record or Export Record after calling Start Import-Export. This direction cannot be changed without stopping the import/export and then starting it again. If you want simultaneously to import lines from one file and process and export them to a different file, then use two Import-Export objects.

Stopping Import-Export

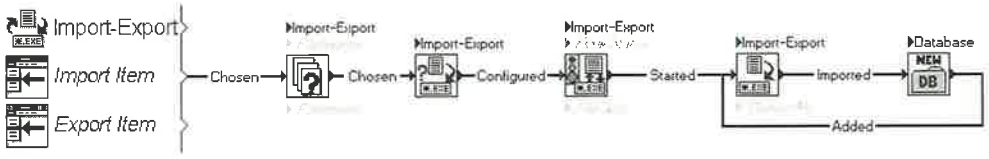
When you're finished importing or exporting, call Stop Import-Export. This closes the text file and the status window (if displayed) and frees the Import-Export object.

With this background, you're ready to start putting import and export function chains together.

Creating an Import Chain

In this section, you'll set up a chain of functions for importing records into the Rolodex database. While the functions needed to support importing may vary from one application to another, this chain is representative of the general case in which you leave it to the user of your application decide exactly how data is to be imported.

You'll set up a function chain to look like the following:

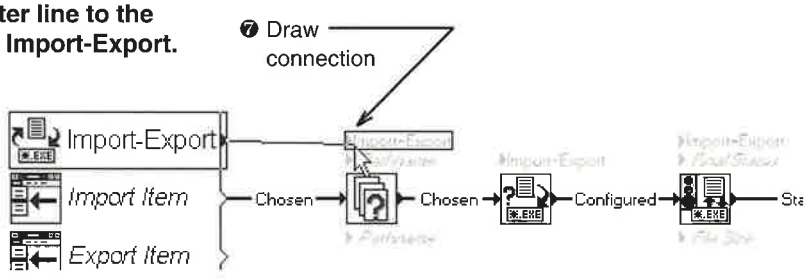


To build this chain:

1. **Select Import-Export in the category list of the Object & Function Palette.**

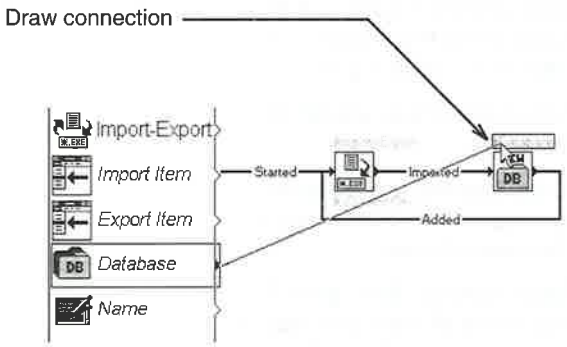
☛ **Note** To select a new function category, you may scroll through the category list or you may click on the bottom of this list and choose the category from the pop-up menu that appears.

2. **Drag from the Palette the following functions and connect them to each other in this order:**
   - **Choose I/E File**
   - **Configure Import**
   - **Start Import-Export**
   - **Import Record.**
3. **Connect the Choose I/E File function to the Import menu item in the object group.**
4. **Select the Database function category.**
5. **Drag Add Record function and connect it to the Import Record function.**
6. **Draw function flow from the right side of the Add Record function to the left side of the Import Record function to form a loop, as shown on the previous page.**
7. **Draw parameter connections from the Import Export object in the object group to each of the Import-Export functions in the import chain. Connect each parameter line to the input labeled Import-Export.**



continued...

8. **Open the Database subject.**
9. **Select the Database object and choose Share from the Object menu.**
10. **Return to the Import-Export subject.**
11. **Choose New Alias from the Object menu.**
12. **Select the Database item in the Subjects list.**
13. **Select the Database object in the object list and click Make Alias, click Close.**
14. **Draw a parameter connection from the Database object alias to the Database input label of the Add Record function.**



Choose I/E File Displays the standard open file dialog allowing the enduser to choose the file to import.

Configure Import Displays the Import Configuration dialog allowing the enduser to choose which fields to import and which objects these fields will be connected to.

Start Import-Export Opens the text file.

Import Record Imports a line of text and dispatches its fields to objects in your application.

Add Record Creates a new record in the database.

☛ **Note** This process of importing a line of text and creating a database record is repeated in a loop (the cycle created by linking the flow out of Add Record to the Import Record function) until there are no more lines to import.

The Import Record-Add Record loop terminates when there are no more records to import, and at that point the import function chain reaches its effective conclusion. However, there is a certain amount of low-level cleanup that has to take place before the import process is completely finished. This cleanup is handled by the Stop Import-Export function.

continued...

This completes the basic function chain for importing records. Following is a brief explanation of what the chain does.

To complete the import function chain, then:

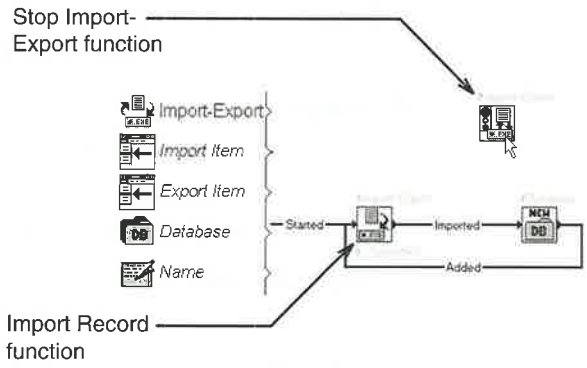
15. Select the Import-Export function category and drag the Stop Import-Export function onto the worksheet. Place it above and to the right of the Import Record function.

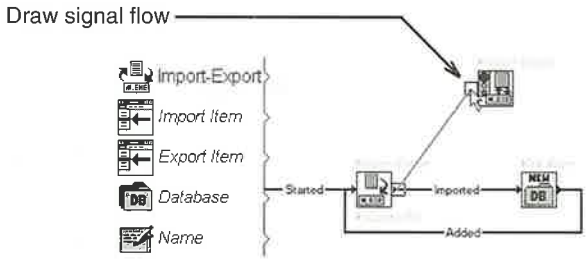
16. Draw a signal flow from the Import Record function to the Stop Import-Export function.

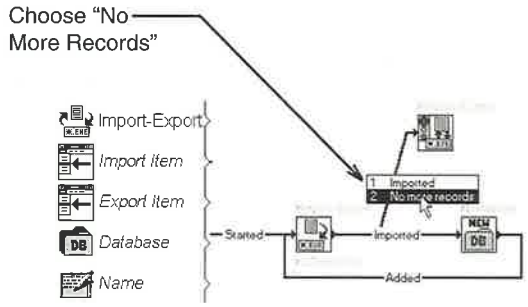
17. Click on the pop-up flow label connecting the two functions and choose the No More Records flow.

When the end of the text file is reached, the Import Record function issues a flow indicating that no more records are available. You use this flow to call Stop Import-Export. This "officially" terminates the import.

18. Connect the Import-Export object to the parameter input of the Stop Import-Export function.







One last bit of housekeeping. It may rarely happen that some sort of file error prevents import from proceeding after the Start Import-Export function has been called. In such an event, import should be officially terminated with Stop Import-Export. You may also want to alert your user that an error has occurred.

To do this:

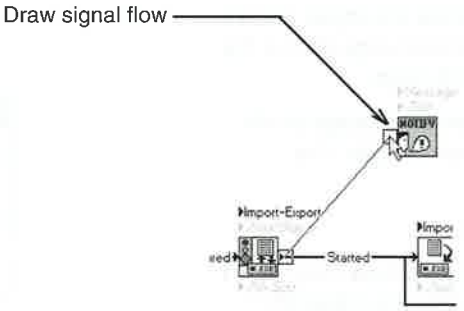
19. Select the Notification function category.

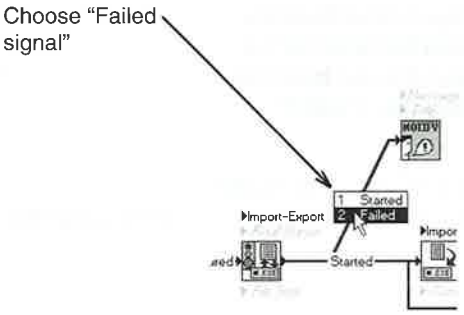
20. Drag the Alert Notify function onto the worksheet. Place it above and to the right of the Start Import-Export function.

Use this function to display a message dialog to alert the user.

21. Connect Alert Notify to the Start Import-Export function.

22. Click on the pop-up flow label connecting the two functions and choose the If error occurs flow.





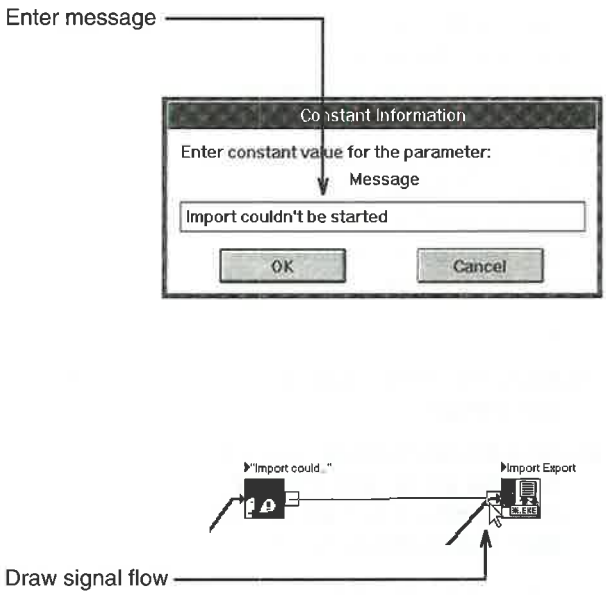
☛ **Note** When you enter the message in this way by directly declaring the text of the message, you're limited to 255 characters. To display longer messages, you can connect Alert Notify's Message input to a Text object. Directly declaring textual inputs is generally used just for single constants, but may, as here, be used to enter somewhat longer fixed text as well.

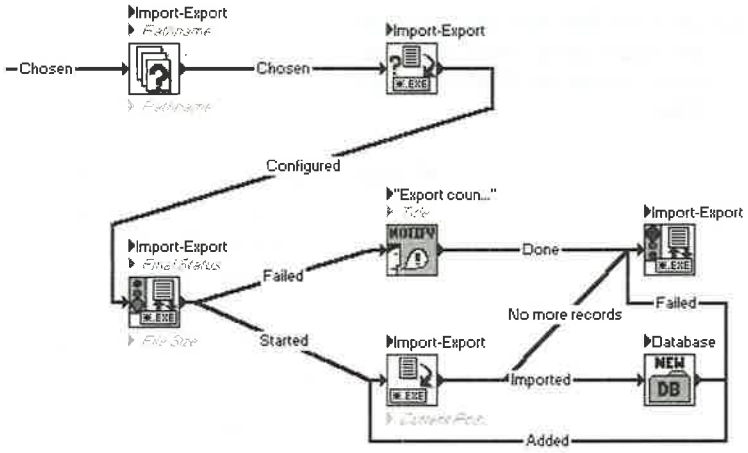
Labels display about a dozen characters only. The remaining characters are displayed as three dots (...). The message itself is unaffected. To view the message, double-click the input label.

continued...

23. **Double-click the input parameter label *Message* above the Alert Notify icon.**
24. **Type a brief message such as, "Import couldn't be started".**
25. **Click OK.**
26. **Draw a signal flow from Alert Notify to the Stop Import-Export function.**
27. **Draw a signal flow from the Add Record function to the Stop Import-Export function and choose the "Failed" signal.**

Your finished function chain should look something like the following diagram.

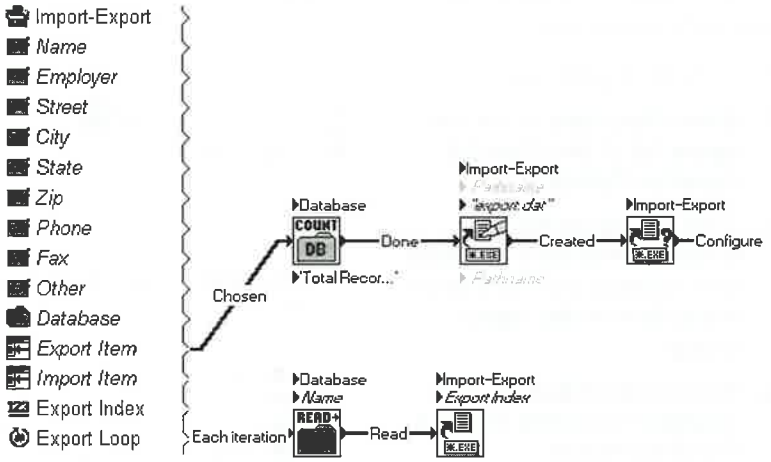




Creating an Export Chain

In this section, you'll set up two related chains of functions for exporting records from the Rolodex database to an ASCII text file. While the functions needed to support exporting may vary from one application to another, these chains are representative of the general case in which you let the user of your application decide exactly how data is to be exported.

The function chains you're going to set up look like this:



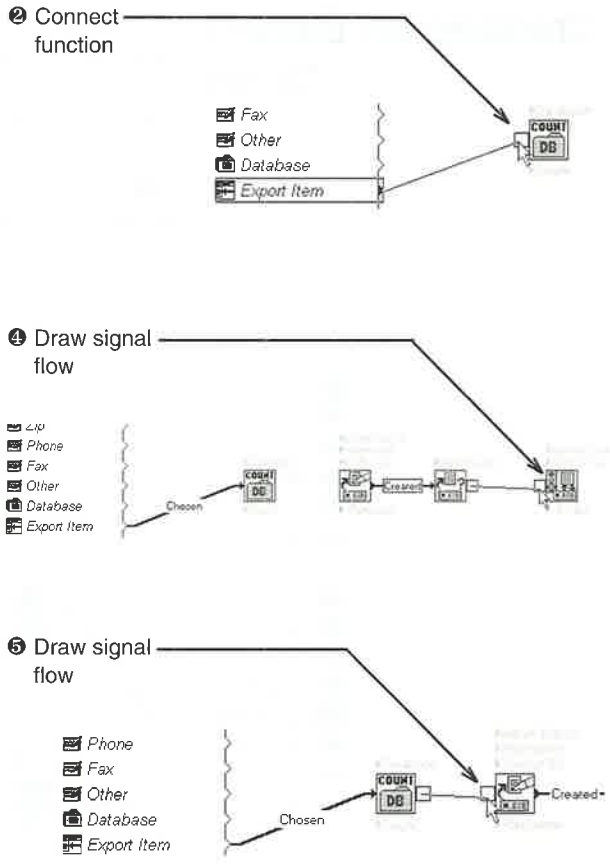
continued...

Before you continue, you may want to arrange the items in the current worksheet to allow you to build the Export function chain above the *Import* function chain you just completed.

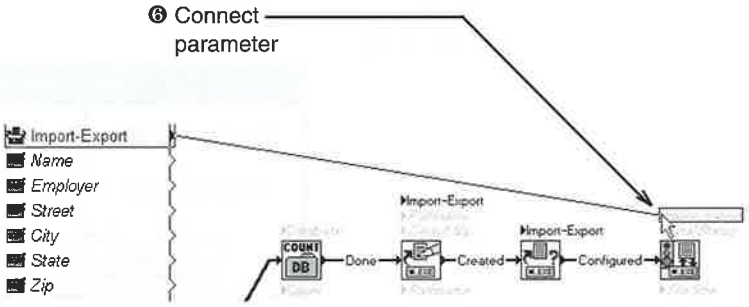
To do this, click in the worksheet and drag a marquee around the object group and the associated function icons. Move the selected items to their new locations and release the mouse button.

To build the Export chain:

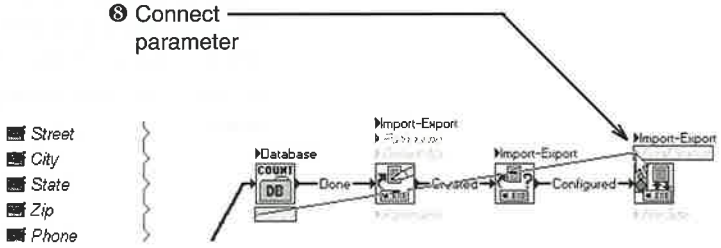
1. **Select Database in the category list of the Object & Function Palette.**
2. **Drag the Count Records function onto the worksheet and connect it to the Export menu item in the object group.**
3. **Select Import-Export in the category list of the Object & Function Palette.**
4. **Drag out the following functions and connect them to each other with signal flows in this order: Create Export File, Configure Export, Start Import-Export.**
5. **Draw a signal flow between Count Records and Create Export File.**



6. **Draw parameter connections from the Import Export object to each of the Import-Export function inputs labeled Import-Export.**



7. **Draw a parameter connection from the Database object alias to the Database input label of Count Records.**
8. **Draw a connection from the Count Records output label to the Final Status input of the Start Import-Export function.**



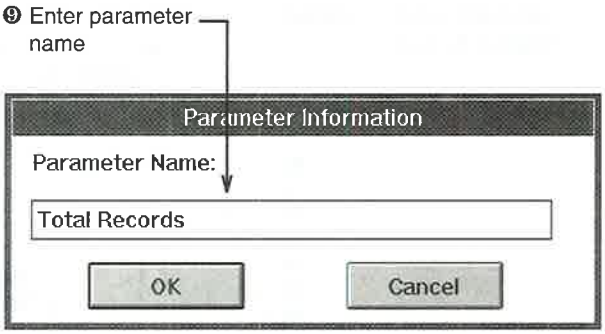
A dialog appears in which you are asked to enter a name for this parameter connection.

continued...

9. **Type a parameter name, such as Total records.**

10. **Click OK.**

continued...



☛ Note The connection you just made between count records and Start Import-Export instructs the Import Export object that it is to export as many records as there are in the database. This information is used in generating the status dialog.

As you can see, it's possible to connect a function output to the input of another function. In fact, you can connect a given output to multiple inputs. When you make such parameter connections between functions, you're instructing AppWare to create a temporary object called a "pseudo-object," (see *Essential ALMs*) that persists in memory as long as it is needed and is then destroyed.

Export Process & Loops

So far, the export function chain is similar to the one you set up for importing. On the Import-Export side, the basic sequence of operations is to create an export file, configure the export, and start the export process. The Count Records function, which has no counterpart in the import process, is used to determine just how many records are to be exported. You'll learn more about its purpose in just a bit.

Like importing, exporting involves a repetitive process or loop that contains both Import-Export and Database functions. During import, you'll recall, a line of text was read into the application and then added to the database file. This process repeats until there are no more lines to import. During export, the process is reversed: a record is retrieved from the database and then exported to a file. The cycle repeats until there are no more records to export.

As with import, you could set up an actual circle of functions to manage the export process. In this case, however, the chain would be a little more complicated since you are now responsible for deciding how many cycles the loop must go through before it terminates. Each time through the loop, a choice must be made either to continue or to terminate. Functions have to be set up to make that choice.

A simpler way to program this and most other loops is to use the Loop object. With the Loop object, you can define an arbitrary function chain as a loop and specify up front the number of cycles through which it will pass before terminating.

To define a function chain as a loop, you attach the chain to a Loop object. To start the Loop, you call the Start Loop function. To specify the length of the loop (the number of cycles), you pass the number of cycles as an input parameter to Start Loop.

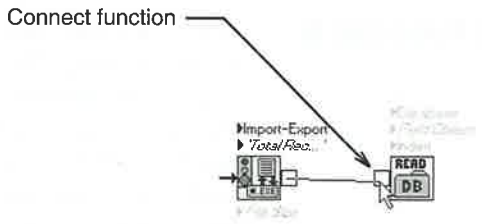
To further simplify your programming task, the Loop object automatically increments a specified Number object so that you know how far along in the process you are. This number often serves as an input parameter to functions inside the loop that use the loop index to retrieve data.

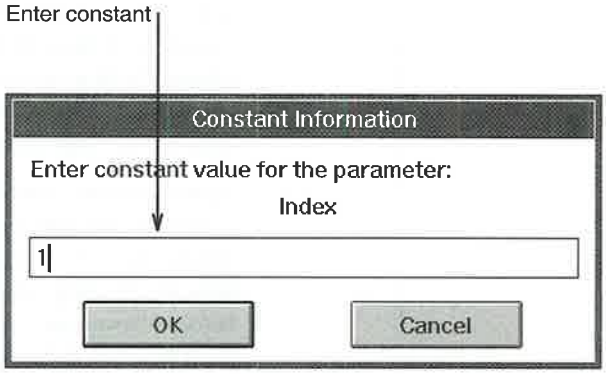
11. **Select Database in the Category list of the Object & Function Palette.**
12. **Drag the Read Record function onto the worksheet and connect it to the Start Import-Export funciton.**
13. **Drag a parameter connection from the Database object alias to the Database input label of Read Record.**
14. **Drag a parameter connection from the Text object called Name to the Field Object input label of Read Record.**
15. **Double-click the Index input label of Read Record and enter "1."**
16. **Click OK.**

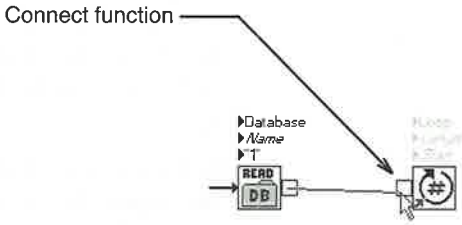
    The best way to learn how the Loop object works is to see it in action. Let's return to the export chain and set up the export loop.

17. **Select the Loop function category in the Object & Function Palette.**
18. **Drag the Start Loop function onto the worksheet and connect it to the Read Record function.**

    At run time, after calling Start Import-Export, the flow of operations passes from this function chain to the loop chain. You'll define this chain later.



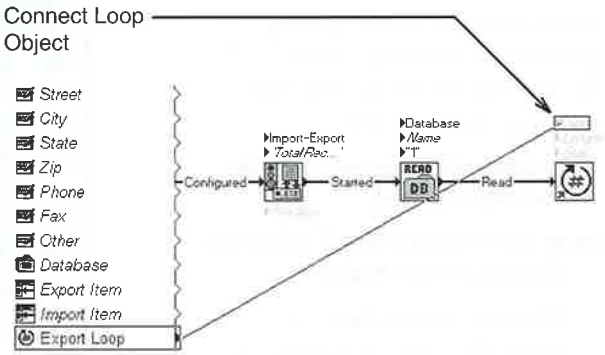




19. Find the Loop object, drag it to the object group. Label it "Export Loop."

20. Connect the Loop object to the Loop input of the Start Loop function.

This connection instructs the Start Loop to trigger the function chain attached to the Loop object.

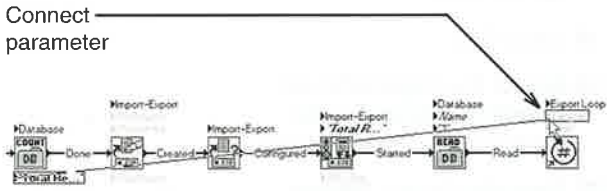


21. Find the Number object in the Object and Function palette, drag it to the object group. Label it "Export Index."

This object serves as the index number that the Loop object increments each time it cycles through its associated function chain. This same number is used as a reference to get indexed records out of the database file.

22. Connect the Count output of the Count Records function to the Length input of the Start Loop function.

Count Records returns the total number of records in the database file, each of which is to be exported. The length of the export loop is therefore equal to the number of records to export. That's what you indicate by making this connection.



continued...

Now it's time to set up the Loop itself. As in the import function chain, so here the loop is a simple sequence of two functions, one for getting information out of the database and the other for transferring that information to the export file.

23. Double-click Export Loop in the object group.

24. In the Assign Value To pop-up choose Export Index.

Choosing Export Index here means that with each cycle, the Loop object increments the value of Export Index by 1. The content of Export Index thus indicates the number of cycles through which the loop has gone. You'll use this index to retrieve successive records from the database.

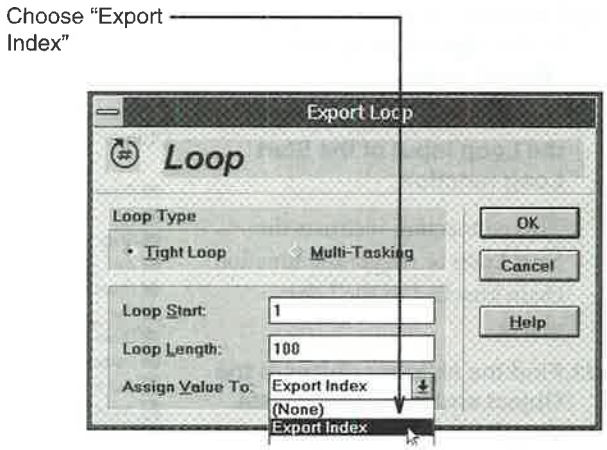
25. Click OK.

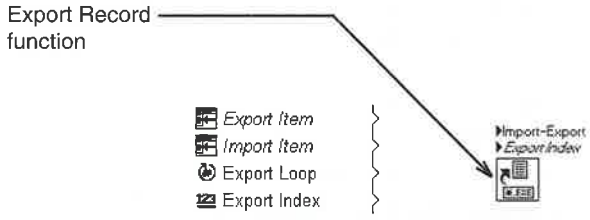
26. Select the Import-Export function category in the Object & Function Palette.

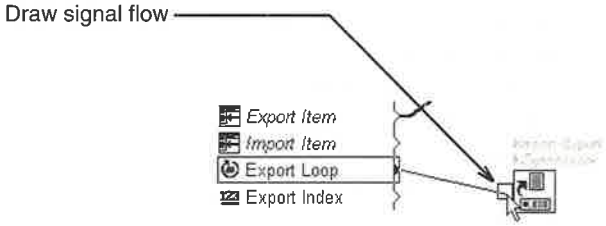
27. Drag the Export Record function onto the worksheet and place it next to the Export Loop object.

28. Draw a signal flow from the Export Loop object to the Export Record function.

29. Select the Database function category in the Object & Function Palette.

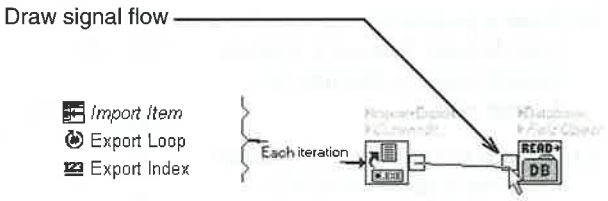






30. Drag the Read Next Record function onto the worksheet and place it to the right of Export Record.

31. Draw a signal flow between the two functions.



☛ Note

Read Next Record is one of several functions used to retrieve database records in indexed order from a database file. Records are retrieved by their index in alphabetical order relative to a particular field. If, for example, you use Name as the Field Object, then Jane would precede John. Jane would have index *n*, John index *n+1*. On the other hand, if John works for Any Computer and Jane for Ernst and Young, and you use Employer as the Field Object, then John's record will precede Jane's when returned by Read Next Record.

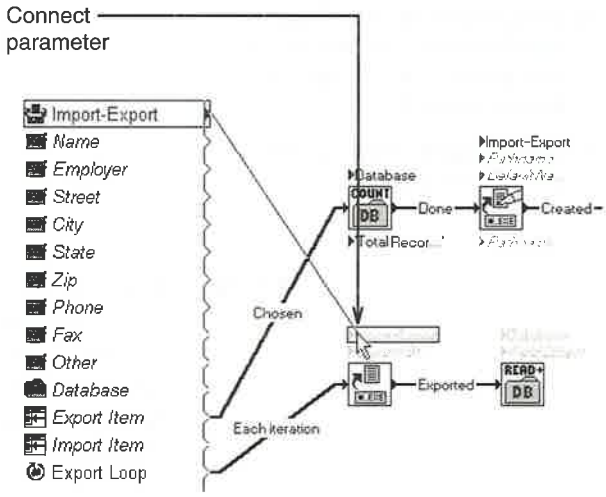
Note the name of the signal issued by the Loop object: Each iteration. The Loop object issues this signal as many times as there are cycles or iterations in the loop. If, for example, you're exporting 50 records, the Loop object will issue 50 of these signals, thereby calling the attached function chain 50 times, once for each record.

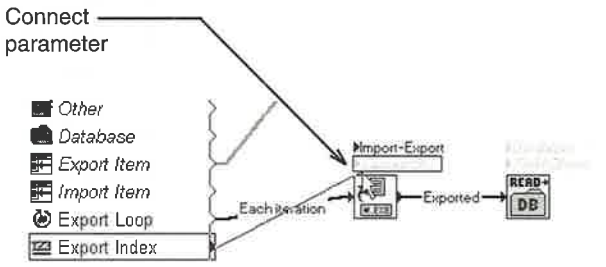
continued...

32. Draw a parameter connection from Export Record's Import-Export input to the Import-Export object.

33. Draw a parameter connection from the Export Record's Current Status input to the Export Index Number object.

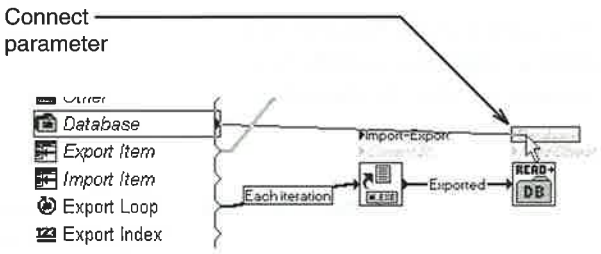
Current Status is used by the Import-Export object to generate a graphical filling bar that indicates the number of records exported relative to the total number of records.



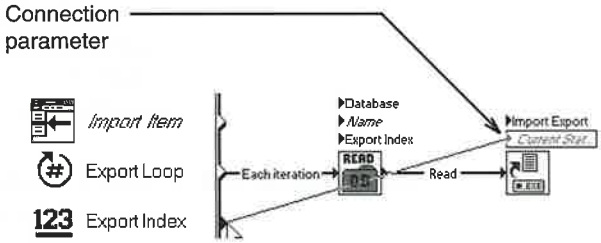


34. Draw a parameter connection from Read Next Record's Database input to the Database object alias.

35. Draw a parameter connection from Read Next Record's Field Object input to the Name object alias.



continued...



This completes the basic function chain for exporting records. Here's a summary of what the chain does.

Count Records Reports the number of records in the database file. The output of this function is used to determined the length of the export loop; that is, the number of records to export. (In this case, it is assumed that you want to export the entire file. This will not always be case, and in more sophisticated programs you may need to use other methods for determining the number of records to export). It is also connected to the Start Import/Export function to used to track the status of the export process.

Create Export File Lets you specify and create a new text file that will contain the records to be exported.

Configure Export Displays the Export Configuration dialog allowing the end-user to choose which fields to export and how they'll be ordered in the export file.

Start Import-Export Opens the text file.

Start Loop Transfers program control to the Loop object, which cycles through its associated function chain as many times as indicated in the Length parameter passed to Start Loop.

Read Next Record Retrieves the next record from the database file and dispatches its data to the appropriate fields in the application.

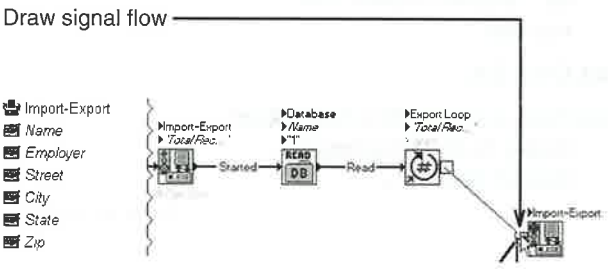
Export Record Exports the current data in the application fields to a line of text in the export file.

This process of retrieving a record and exporting it as a line of text is repeated in a loop until there are no more records to be retrieved.

The Read Next Record-Export Record loop terminates when there are no more records to export, and at that point the export function chain reaches its effective conclusion. As with import, there is still a certain amount of cleanup handled by the Stop Import-Export function that has to take place before the export process is completely finished. To complete the export function chain, then:

36. Draw a signal flow from the Start Loop function to the Stop Import-Export function (at the end of the Import chain).

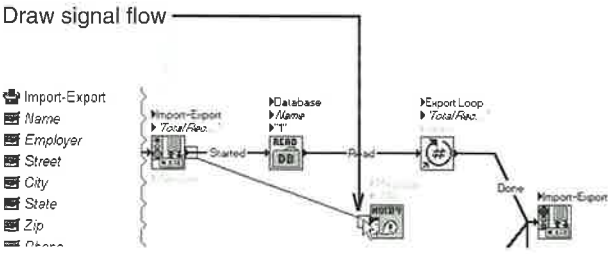
Note the signal name *Done*. In other words, once the Read Next Record-Export Record loop is done, the function chain containing Start Loop picks up where it was interrupted when the loop began.



37. Select the Notification function category.

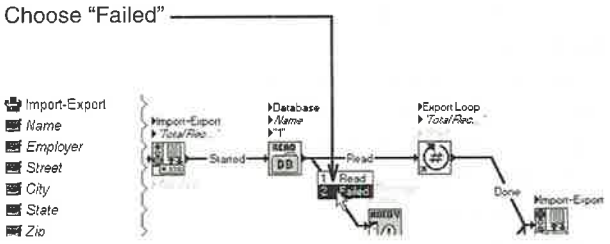
38. Drag the Alert Notify function onto the worksheet. Place it below and to the right of the Start Import-Export function in the export chain.

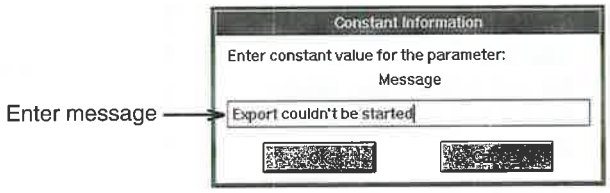
39. Connect Alert Notify to the Start Import-Export function.

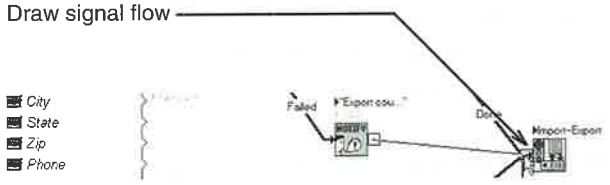


continued...

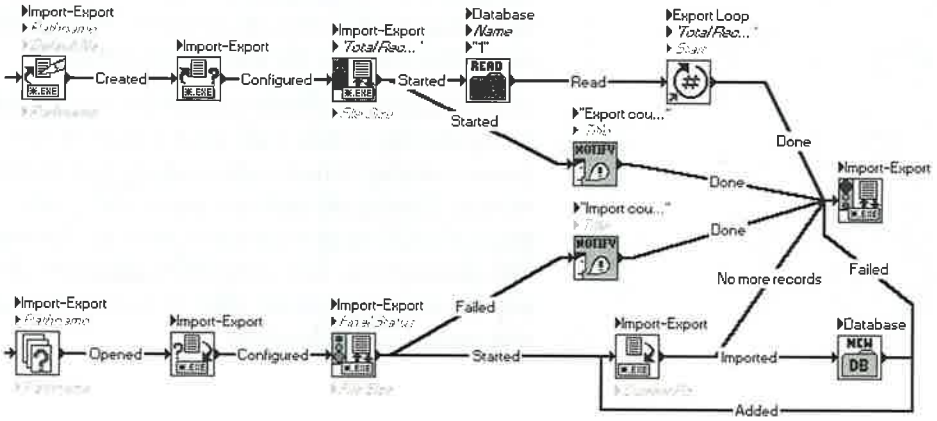
40. **Click on the pop-up flow label connecting the two functions and choose the Failed flow.**
41. **Double-click the message input parameter label above the Alert Notify icon.**
42. **Type a brief message such as, "Export couldn't be started."**
43. **Click OK.**
44. **Draw a signal flow from Alert Notify to the Stop Import-Export function.**







Your finished function chains for importing and exporting should now look something like the diagram below.



Exclusive Mode

The Database object is inherently multiuser. In other words, it permits multiple users on a network to access its data file simultaneously. The Rolodex you're building now, however, is the sort of application you will use for personal record keeping. Multi-user accessibility isn't needed, or perhaps even wanted. If your computer isn't even on a network, it goes without saying that multiuser file access is unnecessary.

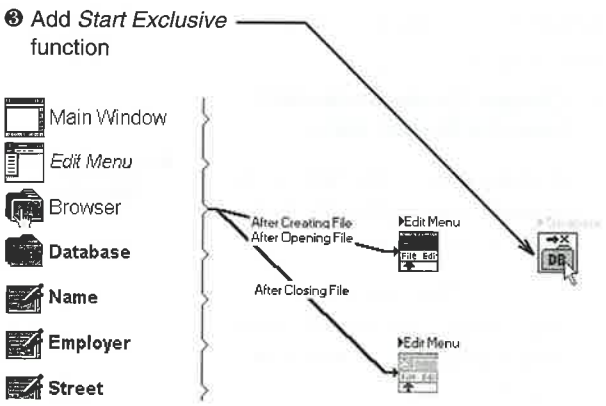
In general, when multiuser accessibility isn't needed, it's a good idea to turn it off, since it tends to slow down certain operations such as adding and updating records. Turning off multiuser access won't make much of a difference when you're working only with individual records. But, in repetitive operations like import and export, the net effect of turning off multi-user access can be substantial.

Turning Off Multi-User Access

To turn off multiuser access, you call the *start exclusive* Database function. This prevents other users on a network from accessing a given file until you specify otherwise. Even if you're not on a network, using this function improves performance. In such cases, Start Exclusive should be called as soon as a file is opened or created.

To make the Rolodex single-user only:

1. **Open the Database subject.**
2. **Select the Database function category in the Object & Function Palette.**
3. **Drag the Start Exclusive function into the worksheet, placing it next to the Enable Menu function.**
4. **Draw a signal flow from Enable Menu to Start Exclusive.**
5. **Draw a parameter line from the Database object in the object group to Start Exclusive's "Database" input parameter.**



☛ **Note** The parameter connection you've just made indicates that Start Exclusive will operate on all data files created or opened by this particular Database object. As your project stands now, an affected file will remain in exclusive mode until it is closed or until you exit the Rolodex application.

You can return a file to multiuser status by calling Stop Exclusive. Further information on this function is available in *Essential ALMs*.

Compiling Your Application

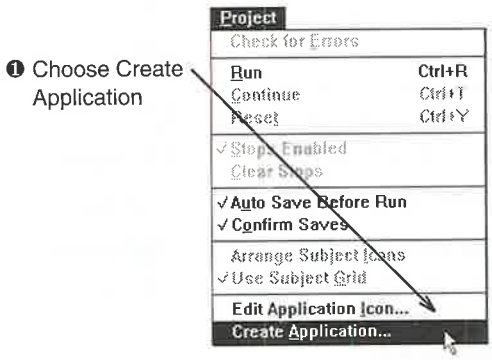
Now that you're finished with Lesson 2, you're ready to compile your program.

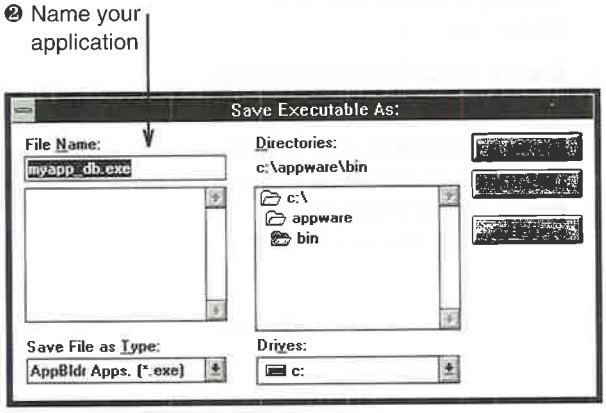
1. **Choose Create Application from the Project menu.**

   A dialog appears prompting you to name your application and locate it on your disk.

   In this dialog, you can give your application a name and choose which directory in which to save it.

2. **Name your application, choose a directory to save it to, and click OK.**



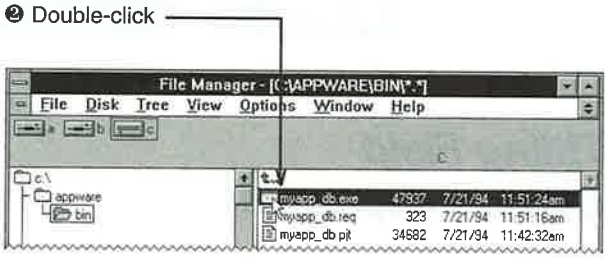


Running Your Application

After AppWare has finished building your application:

1. **Exit from AppWare or go to the File Manager.**
2. **Find your application and double-click its icon in the File Manager.**

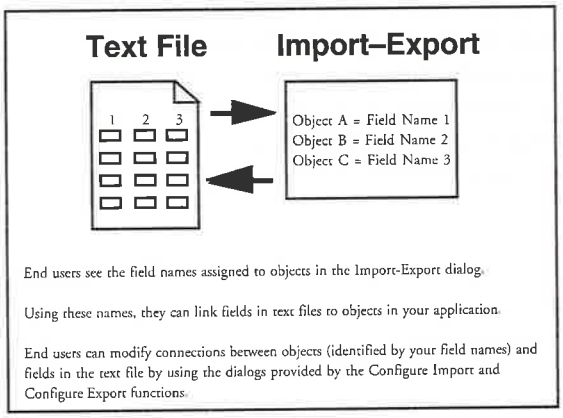
If the application doesn't work quite right, reopen the project and review your work. Make sure functions and their parameters are set up as outlined in this lesson. Recompile and try it again.



IMPORT-EXPORT OPTIONS

Titling Field Objects (Optional)

Once you've added objects to Import-Export, you can assign them run time names. These are the names that the enduser of your application sees at run time in the Configure Import and Configure Export dialogs (explained below). These names are also optionally exportable as the first line of an export file.



To assign run time names to the fields contained in the Import-Export object group:

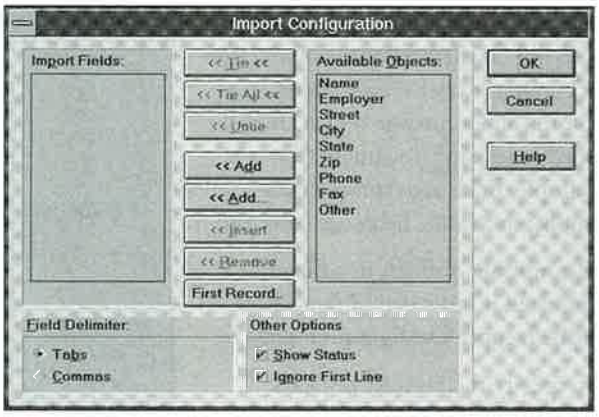
1. **Individually select the objects in the Objects in the Import/ Export: list and enter names for them in the box entitled Field Name. If you wish, you may use the default names.**

   The default name displayed in the Field Name box is the title of the object as it appears in the object group, but you may enter any name you like. Every object in the list must have a run time name, if you plan to allow the end-user of your application to configure import or export.

Configuring Import (Optional)

1. Click the Import... button.

The Import Configuration dialog appears.



Using this dialog, you or the user of your application can determine in detail how the import process works. You can completely configure importing, or you may leave it to the user of your application to do so. If you do it yourself, you should proceed through each of the following steps. At run time, importing can then be implemented without the user having access to this dialog. If, however, the end-user is to have ultimate control over importing, then he or she must be given access to this same dialog. Do this by calling the Configure Import function.

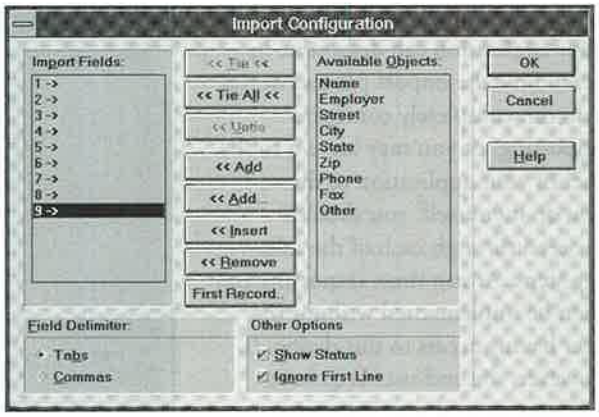
To configure the import process, you or your end-user must:

- Specify the file format
- Match fields in the file with objects in the application.

Specifying the File Format

Before data from a text file can be imported, the Import-Export object needs to know the maximum number of fields there are in a given line of the file. A field consists of a string of characters between two tabs or commas or between a tab and a carriage return. In a word processing document, for example, fields are typically paragraphs; in a table, fields are columns.

The number of fields in a file to be imported is indicated in the left list of the Import Configuration dialog.



To specify the number of fields:

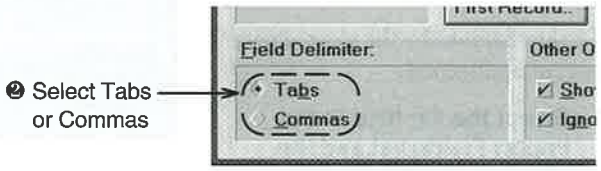
1. **Using the Add or Add... buttons, add as many fields to the Import Fields: list as there are fields the file you're updating.**

If lines in the file to be imported contained three fields, Name, Address, and Phone, you would add three fields to the list. Each time you add a field, a blank, numbered line is appended to the list. The Add button adds fields one at a time, whereas Add... brings up a small dialog allowing you to enter the number of fields all at once. A maximum of 255 fields is permitted.

To remove a field, select it and click the Remove button.

In a Text file, fields can be delimited by either tab or comma characters. You must select one of these formats.

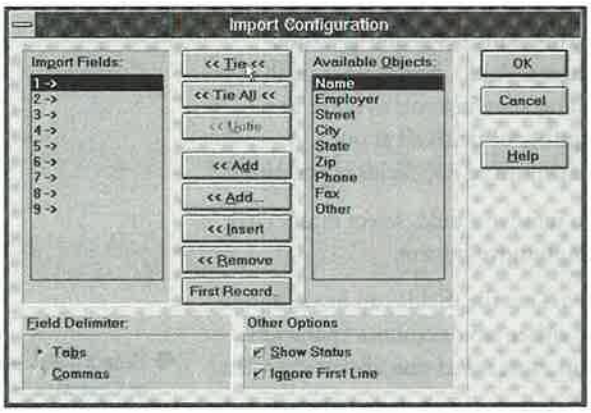
2. Select either Tabs or Commas in the Delimiter box.



Matching Objects to Fields

The next step in the configuration process involves linking file fields to objects in your application.

At run time, when a line of text is imported from a file, its fields, consisting of strings of tab- or comma- delimited characters, are transferred to objects in your program "tied" to those fields.



To tie a field in the file to an object in your application:

1. **Select the file field in the Import Fields list and the object in the Application Objects list, then click the Tie button.**

   Alternatively, you can select the file field, then double-click the object, or select the object first and double-click the file field.

To tie all the elements of one list to the corresponding elements in the other, use the Tie All button. If the two lists are of unequal length, elements at the end of the longer list that have no counterpart in the shorter are left untied. For example, if one list contains three elements and the other five, then clicking «Tie All« would tie Field 1 to the first object, Field 2 to the second object, and Field 3 to the third, but elements four and five of the longer list would remain unconnected.

Not all fields in the import file need be linked to objects in your application. If there are fields you want to ignore, simply leave them untied.

To untie an object connected to a file field:

1. **Select the tied field in the Import Fields list and click the Untie button, or simply double-click the selection.**

Importing Field Names

When database applications export their data to text files, field names are often written as the first line.

You'll generally want to skip over this line and not treat it as data. By default, the Import-Export object starts with the second line of the text file. If you want to consider the first line as data and not skip it:

1. **Uncheck the Ignore First Line checkbox.**

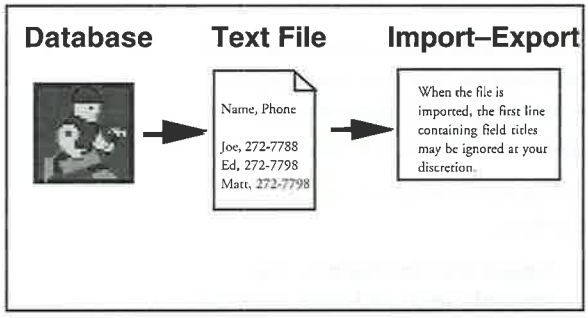
To label objects in the Import-Export Objects list with the field names found in the first line of a text file:

1. **Click the First Record... button.**
2. **Open the text file of your choice.**

   The fields in the first line are displayed in order next to their index numbers in the left list.

For example, if the fields in the first line of the file were Name and Phone, then Field 1 in the Import Fields list would be titled Name, and Field 2 Phone. This convenience is in effect only while the Import-Export object dialog is open.

First Record works only when the file is closed. If you're currently editing the file, close it before clicking First Record.



Displaying Status Dialog at Run Time

By default, the Import-Export object displays a status or progress dialog during the import process at run time. The dialog appears when Import Record is called and is displayed until Stop Import-Export is executed. The progression of the import process is indicated as a filling bar graph.

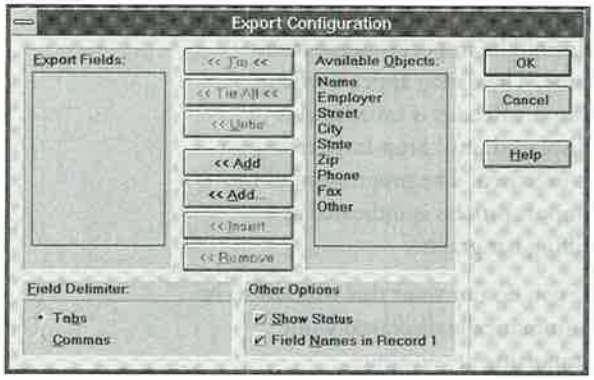
If you do not want this window to be automatically displayed while importing at run time:

1. **Deselect the Show Status checkbox.**

Configuring Export (Optional)

1. **Click the Configure Export... button.**

   The Export Configuration dialog appears.



Like Import Configuration, the Export Configuration dialog allows you to tie objects in your application to fields in the text file. As before, you specify the file format by indicating the number of fields in the export file and the means of their delimitation (tabs or commas). You may also choose whether to export field names (as the first line in the file) and whether to display the status dialog.

3

Using AppWare

BEFORE YOU START

Designing an Application

The way you program a project largely depends on the sort of user interface you want your application to have. Before investing too much effort in actual programming, you should have a good idea of how the user will interact with your application.

Use the Window object's drawing tools, or, if you prefer, a painting or drawing program to design a basic user interface. It's not important at this stage that your design be pixel-perfect. What is important is that you get an idea of the objects you'll need, where they'll be located, and how they'll interact with one another. An hour spent in drawing user interfaces will save you hours of configuring and reconfiguring objects, designing and redesigning function chains.

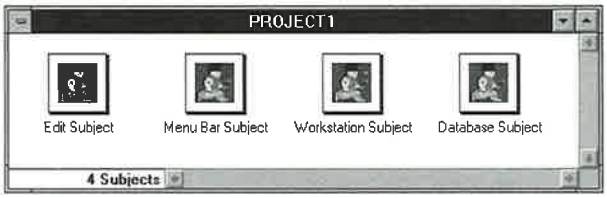
Projects

The first step in building a new application is to create a *project.* This contains your application code organized in subprojects or *subjects.* Although much easier to create and understand, a project is like the source code file a programmer creates in C or Pascal. After you have finished constructing your project, you compile it to create an application (.EXE), just as you would a C or Pascal project, that is accessible by double clicking.

The project file is therefore a container for the subjects. It's in the subjects that all of your program logic is located.

Projects can also be used as *subject libraries* to store subjects you may want to reuse in other applications. Storing subjects in a subject library is analogous to storing drawing objects in a drawing library. A project used in this fashion, of course, shouldn't be compiled into an application.

There is no limit on the number of subjects a project can contain; in practice, however, this number is usually less than 50.



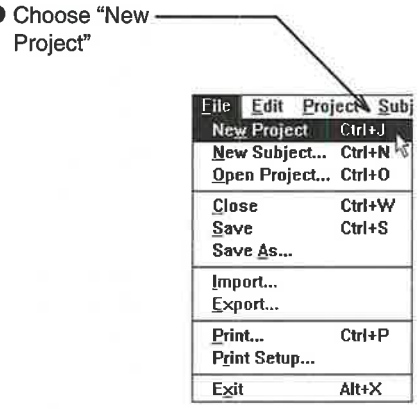
Creating a New Project

1. Choose New Project from the File menu or press Ctrl+J.

A numbered project window appears at the top of the screen, just under the menu bar. For each open project, there is a project window. If multiple projects are open at the same time, their windows are stacked on top of each other. You name a project when you save it.

You can have as many open projects as RAM allows.

All open projects are listed in the order in which they were created in the Windows menu. They are also listed in order of creation in the Navigator.



Opening an Existing Project

1. Choose Open Project from the File menu or press Ctrl+O.

The Open dialog appears prompting you for the file name and the location of the project. You can have as many open projects as memory allows.

Saving a Project

There are two ways to save a project.

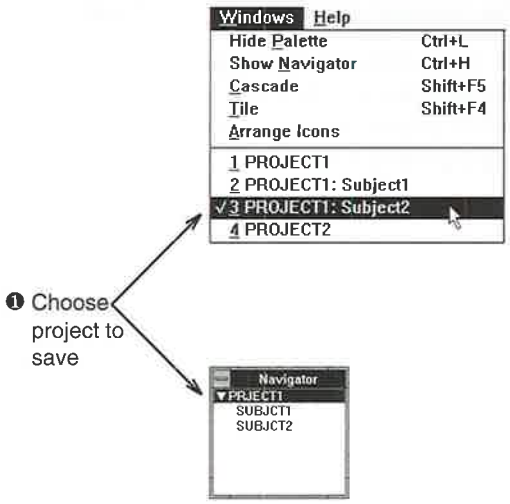
Method 1:

1. **Bring the project window to the front.**

   If you have more than one project open, bring the project you want to save to the front by (1) clicking in the project window, (2) choosing the project name from the Windows menu, or (3) double-clicking the project name in the Navigator window.

2. **Choose Save from the File menu.**

   If this is the first time you've saved the project, you will be prompted to name it and to choose the directory in which it is to be located.



Method 2:

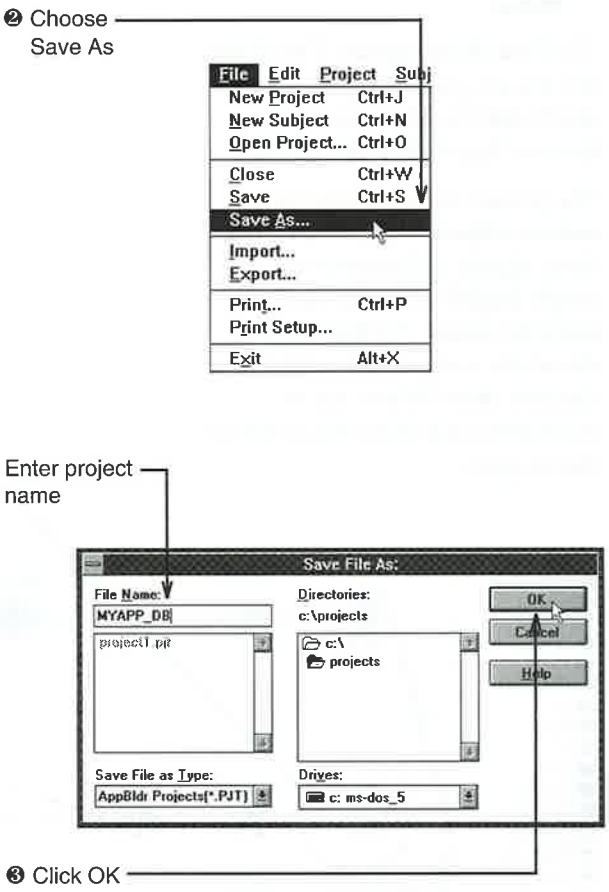
When you choose Save, the project owning the front window is saved in any event. The project window needn't be in front as long as one of the project's subject windows is.

Saving a Project Under a New Name

When you're developing a significant application, it's a good idea to create an archive in which successive versions of the project are saved under different names. This sort of archive will help you avoid having to reconstruct an application entirely from scratch in the event the current project file is ever damaged.

1. **Bring the project to the front, as described in Saving a Project.**
2. **Choose Save As from the File menu and enter the name of the new version.**
3. **Click Ok.**

This creates a copy of the project under the new name and makes the copy the current project. The project under the old name is closed and remains unaffected.



☛ **Note** When you save a project a second time, AppWare creates a backup of the original and names it with a .BAK extension. Subsequent saves follow the same pattern so you always have an older version as a backup.

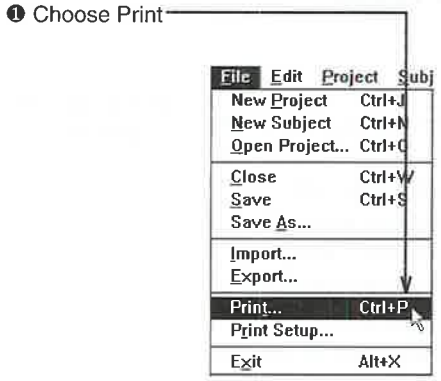
Printing a Project

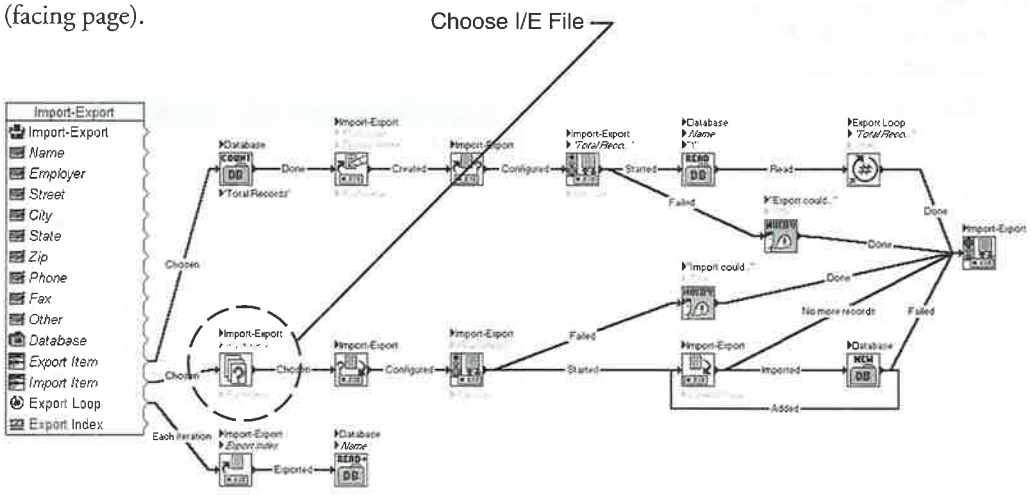
To print a copy of your project:

1. **Choose Print from the File menu.**

The Print dialog appears. This dialog also lets you print your project to an ASCII disk file when you use the Generic/Text Only printer driver.

The printed version of your project reduces subjects, objects, functions, flows, signals, and parameters to simple English script that can be useful for review. To illustrate how this works, compare this subject's function chain (below) and its accompanying printed representation (facing page).





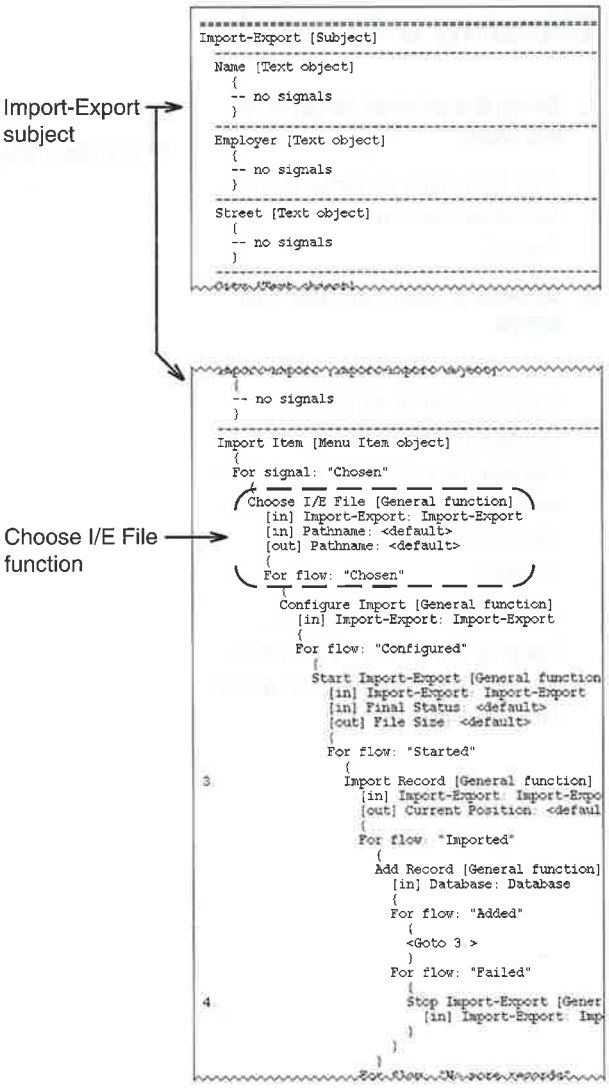
The Import-Export subject's printed representation of the function chain appears at right. The Choose File function that starts the import chain is represented as follows:

Choose File [Import-Export function]
[in] Import-Export:Import Export
[in] Pathname: <default>
[out] Pathname: <default>

Choose File is the function name. The bracketed phrase *Import-Export function* indicates that Choose File belongs to the Import-Export function category.

Function names are followed by the names of their input and output parameters. These in turn are followed by the names of signal flows. The line [in] Import-Export: Import-Export, for example, indicates that an input parameter labeled Import-Export is connected to an object titled Import-Export. The input and output pathname parameters are unconnected, so the function defaults to some predefined behavior. Take a minute to compare the rest of the printed version with the project displayed on the previous page.

This subject is discussed in greater detail in Lesson 2 of the Tutorial.



Closing a Project

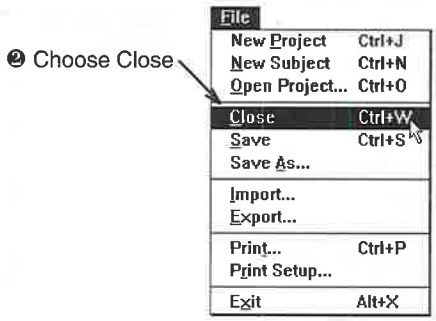
1. **Bring the project window to the front.**

   The three ways of doing this are described above under Saving a Project.

2. **Choose Close from the File menu.**

   The project file, the project window, and all its subject windows will be closed. If changes were made to the project since it was last saved, AppWare asks if you want to save the changes.

   If you choose Close without bringing the project window to the front, you simply close (hide) the front window.



PORTING BETWEEN PLATFORMS

UPSF Files

AppWare incorporates a new technology developed by Novell, called the Universal Program Structure File (UPSF) format. UPSF allows projects created in AppWare for Windows to be ported to operate on Macintosh computers. Projects you create using AppWare for Macintosh can be ported to Windows as well. In order for a project to be ported in its entirety, the objects and functions it uses must be available on both platforms.

UPSF format is a file specification similar in concept to an Adobe PostScript file. Many different printers and typesetters understand and print PostScript files. Likewise, the Macintosh and Windows versions of AppWare speak a common language in UPSF format. Over time, Novell will introduce AppWare for several additional platforms, making your existing and new AppWare-built applications portable to those platforms.

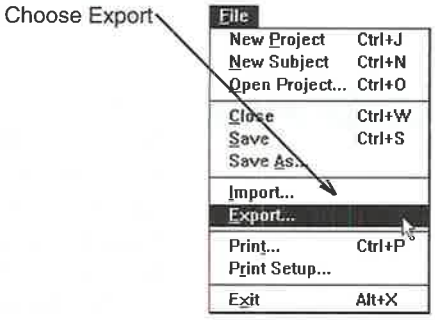
☛ **Note** An project will port using the UPSF format, even if certain objects aren't cross-platform. Those projects which contain platform specific objects will need some editing before the can be executed.

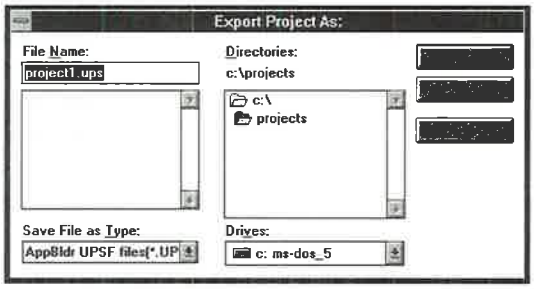
Porting: Windows to Macintosh

When you want to port a Windows application to Macintosh:

1. **Export a UPSF file from your Windows project by choosing Export from the File menu.**
2. **Move the UPSF file to a Macintosh by network or floppy disk. (When using floppy disks, you must use floppy disk exchange software such as Apple File Exchange or Access PC.)**
3. **Launch AppWare for the Macintosh. Import the UPSF file into the Macintosh version of AppWare by choosing Import from the File menu.**

   In the import process, your Windows project is reconstructed on the Macintosh to work just as it did on Windows. You can send projects back and forth between the platforms using the UPSF mechanism.





Porting: Macintosh to Windows

1. **Open your project in AppWare for Macintosh.**
2. **Choose Export from the File menu.**
3. **Move the file to a Windows-based computer using a network or floppy disk (you must use floppy disk exchange software, such as Apple File Exchange or Access PC when using floppy disks).**
4. **Launch AppWare for Windows**
5. **Choose Import from the File menuand open the UPSF file.**

Errors in UPSF Ports

During the import process, a log file is created in the UPSF file's directory recording any errors encountered along the way. A UPSF error occurs when you port a project that uses objects or functions that are not available on the destination platform. For example, if you use the XCMD object in the Macintosh version of AppWare, you will not be able to completely port the project to AppWare for Windows, since such an object is not available for Windows. Keep this in mind as you create applications for either platform.

Undoing

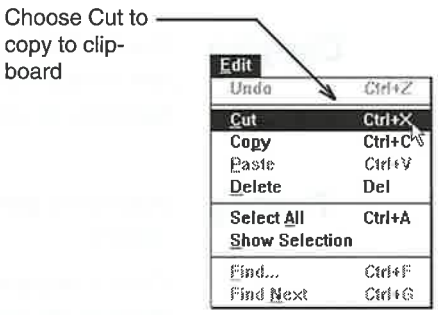
To undo an action:

1. **Choose Undo from the Edit menu or press Ctrl+Z.**

   AppWare supports a single level of undo for most operations, including the deletion of subjects.

Using the Clipboard

All components in a project — subjects, objects, object aliases, and functions — can be cut and copied to AppWare's internal clipboard and pasted from it. You can use copy and paste, for example, to create multiple copies of an object in an object group without having to drag each instance from the Object & Function Palette. You can also copy components between subjects. Note, however, that when copying an object containing or referencing other objects (containing objects include Database, Group, Menu, Table, Window), you must copy the contained or referenced objects (Text, Picture, etc.) as well.



☛ Note Since the clipboard used for cut, copy, and paste is internal to AppWare, its contents are erased when you exit AppWare. This information applies only to the Windows version of AppWare. On Macintosh, the standard Clipboard is used.

continued . . .

Cutting

When you cut a function, all connected parameter links are deleted; only the function itself is stored on the clipboard. Flows between functions you cut are also severed.

Copying

Parameter links connected to functions, however, are not maintained in the copy. Links between copied functions are also not stored on the clipboard.

Pasting

Subjects are pasted into the current (front) project window.

Objects are pasted into the currently selected object group of the current subject. If no object group is selected, a new one is created. Functions and function chains are pasted into the subject worksheet of the current subject. If no subject is open, the Paste menu item is disabled.

Deleting

1. **Select the component(s) you want to delete, then choose Delete from the Edit menu or press the Delete key.**

   The selected item(s) in the active window will be deleted.

   Before deleting a subject from the project window that contains shared objects, AppWare alerts you to confirm the deletion.

It is possible to reestablish a link using a dummy alias by:

1. **Hold the Alt key and drag the desired object onto the worksheet.**

   A dummy alias of the object appears.

   Make sure the dummy alias is the same *type* as the shared object.

2. **Title the dummy alias with the same name as the shared object.**

   The links between shared objects and object aliases depend on the alias name and object type.

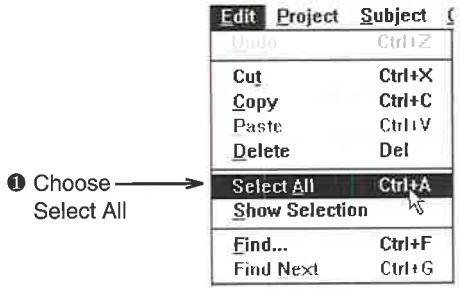
☛ **Note** Deleting a subject with shared objects also deletes the links between the shared objects and their aliased objects. This particular operation cannot be undone.

Selecting All Subject Items

To select all items in the current (front most) subject:

1. **Choose Select All from the Edit menu.**

   All items in the current subject are selected (appear highlighted).

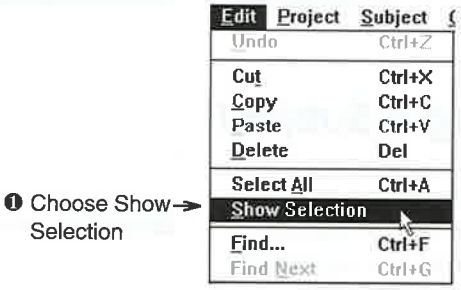


Showing Selected Items

To show all selected items in the current (front most) subject:

1. **Choose Show Selection from the Edit menu.**

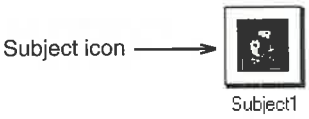
   The selected items in the current subject are centered in the subject worksheet.



SUBJECTS

Creating a Subject

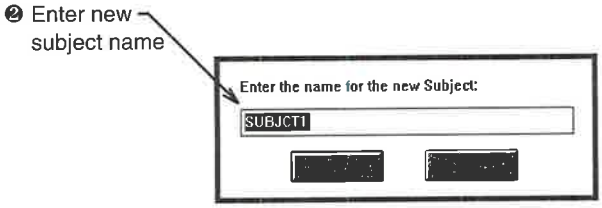
As easy as an AppWare application project is to recreate, it represents programming that you would rather not redo. So, AppWare lets you group objects and functions into reusable subprojects, or *subjects*.



1. **Open the project that will contain the new subject.**

   Subjects are always contained in a project. You can create a subject only when a project is open.

2. **Choose New Subject from the File menu. When prompted, enter a name for the subject.**



☛ Note A subject icon is displayed in the active project window with the name you gave it, and a new subject window is opened. This window is entitled as follows:

Project Name:Subject Name

For example, if you made Project1 the active project, and created a new subject, the new subject icon in Project1 would be labeled Subject 1 and the new subject window would be labeled Project1:Subject1. This name also appears in the Windows menu.

You can have as many open subjects in as many open projects as you want. You're limited only by the amount of RAM in your machine.

Opening and Closing Subject Windows

To open a subject window, you may use any of the following methods:

Method 1:

1. **Open the project window containing the subject you wish to open.**
2. **Double-click the subject icon in the project window.**

The subject window opens up across the middle of the screen, or wherever you last left it.

Method 2:

1. **Choose the subject from the Windows menu.**

   The subject must be open and in the background.

Method 3:

1. **Double-click the name of the project in the Navigator.**

   This will open the project window.
2. **Double-click the subject icon in the project window..**

☛ **Note** Remember that subjects are not independent files. They are contained within project files. To open a subject, you must first open its associated project file. You save subjects by saving the project which contains them. See Saving a Project.

continued...

To close a subject window, you may use any of the following methods:

Method 1:

1. **Double-click the subject's control-menu box.**

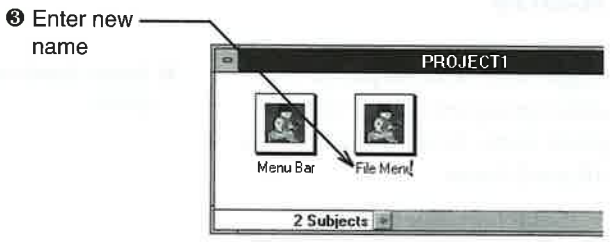
Method 2:

1. **Choose Close from the File menu.**

Remember, closing a subject window does not close the project. The subject window is merely hidden.

Renaming a Subject

1. **Click in the subject's title area in the project window to position the I-beam.**
2. **Drag the I-beam over the text you want to retitle or double-click the title area to select it in its entirety.**
3. **Type the new name.**

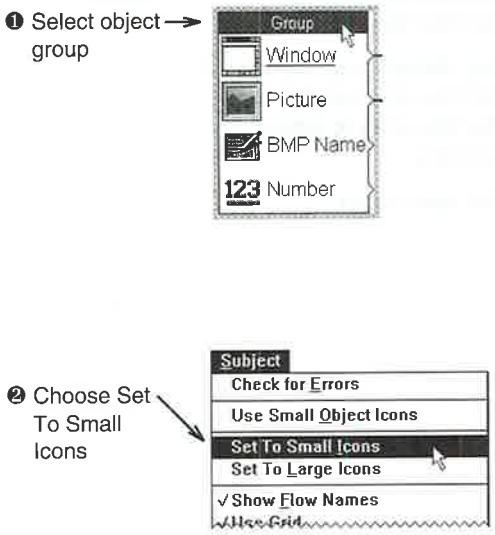


Displaying Large & Small Object Icons

Large object icons displayed in the object group are full-size (32 x 32 pixel) icons. To show smaller (16 x 16 pixel) icons:

1. **Select an object group.**
2. **Choose Set To Small Object Icons in the Subject menu.**

   All icons in the selected object group are reduced in size.

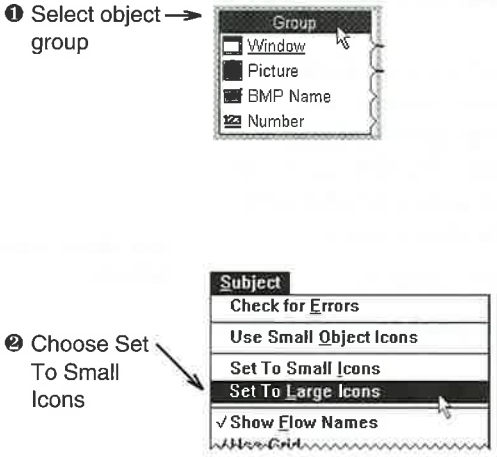


To show large object icons:

1. **Select an object group.**
2. **Choose Set To Large Object Icons in the Subject menu.**

   Object icons in the selected object group appear as large.

The choice of large or small object icons is specific to a given group. You may mix the use of small and large object icons within a subject.



☛ **Note** You can also configure how object icons will appear in an object group, before dragging them onto the worksheet. To cause objects to appear as large (32 x 32) icons in future object groups, choose (uncheck) Use Small Object Icons in the Subject menu. To cause objects to appear as small icons in future object groups, choose (check) Use Small Object Icons.

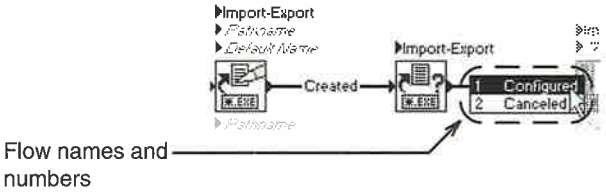
Showing Flow Names & Numbers

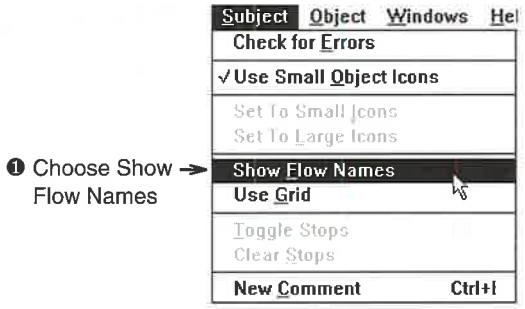
Flow names between functions are informational. However, they occupy a large amount of space in the worksheet. The flow names can be displayed as numbers that correspond to the chosen flows.

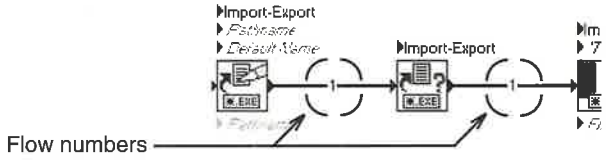
To show flow numbers:

1. **Choose (uncheck) Show Flow Names.**

   The flow names in the current subject appear as numbers corresponding to the chosen flow.







☛ **Note** The Show Flow Names command changes the display of flows between functions only. The flows from objects to functions do not change.

Worksheet Grid

The subject worksheet incorporates an invisible grid to help you align functions. When the grid is activated, groups and function icons placed on the worksheet snap to the nearest grid lines. Icons align themselves along their top and left edges. Activating the grid does not realign icons *already* on the worksheet.

To activate the grid:

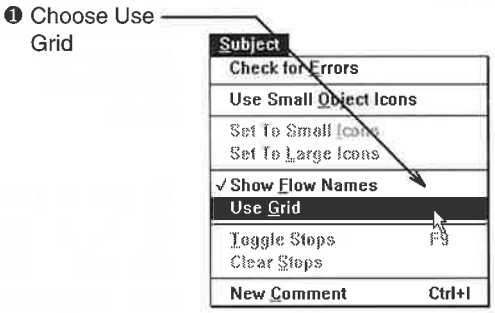
1. **Choose (check) Use Grid in the Subject menu.**

   When first selected, the item is checked and the grid is activated.

To deactivate the grid:

1. **Choose (uncheck) Use Grid.**

Use of the grid is a feature of the subject worksheet. You can use it on a subject-by-subject basis.

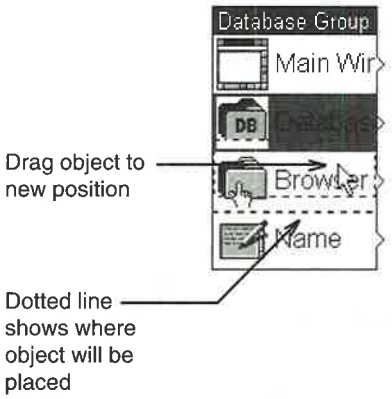


☛ **Note** You can align all functions to the grid by selecting them (Ctrl+A) and moving the grouped selection just a pixel or so. Functions already on the grid will be unaffected; functions not yet aligned to the grid will be repositioned.

Reordering Objects in the Object Group

1. **Use the cursor to select and drag the object(s) up or down to the desired position in the object group.**

   As you move the cursor, a dotted line appears to show you where the object will be inserted in the object group.



☛ **Note** An object's position in the object group does not affect the operation of the resulting application. You'll probably find it convenient, however, to group associated objects together.

Adding Comments

Comments, identified by balloon icons, can be placed anywhere on your subject worksheet.

Comments have two parts, the brief comment and the full comment. The brief comment identifies the comment in the subject window. Brief comments are also listed in the Navigator under their subject in the order in which you created them.

To add a comment to a function chain, do the following.

1. **Choose New Comment from the Subject menu.**

   This displays the comment dialog box.

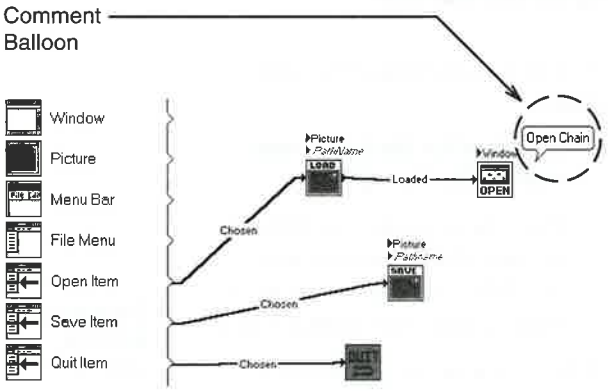
2. **Name the comment in the Brief Comment field.**

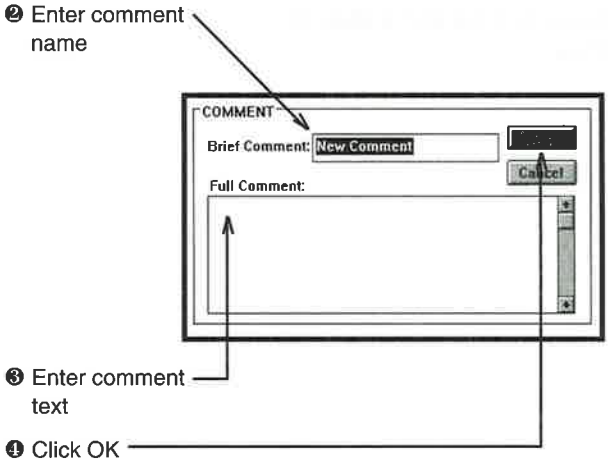
3. **Enter the comment text in the Full Comment field.**

4. **Click OK.**

   A new comment icon appears in the center of the worksheet.

5. **Drag the comment icon to the desired part of the worksheet. You may move it about freely and locate it wherever you wish.**



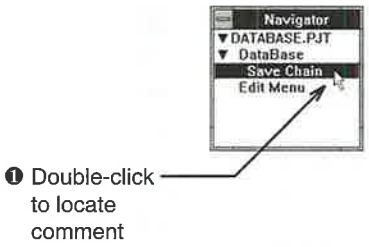


Finding & Editing Comments

To locate a commented function chain:

1. **Double-click the brief comment in the Navigator.**

   The subject window scrolls to the location of the comment icon and flashes the comment icon in order to draw your attention to it.



When you want to read or edit a comment:

1. **Double-click the comment icon.**

You can also search for comments using the Find dialog (Edit menu: Find . . .).

Sharing & Aliasing Objects

On the function worksheet, you use lines to link functions to objects and to other functions. This methodology works fine within a subject. However, it can't be used between subjects since each is its own separate window and lines can't be drawn across window boundaries. When a function or object in Subject A needs to reference an object in Subject B, you create a surrogate or *alias* for the object in B and put it into A. Objects and functions in A work with the alias just as they would with the original, which still resides in B. Object aliases are, therefore, the links that hold subjects together.

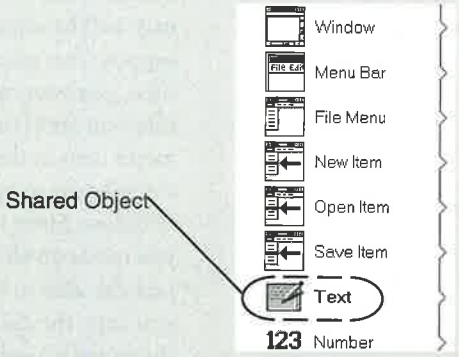
An object alias is not a copy of the original object but a representative of it that you can use as if it were the original. The difference between a copy and an alias is easily illustrated by an example. Let's suppose that you're working on a project that contains two subjects, one devoted to a menu bar and its constituents, and another concerned with windows. While the responsibilities of the two subjects are generally distinct, there may well be interaction between them. For instance, suppose that when you choose a menu item at run time, you want to open a dialog window. To program this, you attach an Open Window function to the menu item in the menu bar subject. Next, you specify the window to open as the input parameter to Open Window. Since that window is in the windows subject, you create an alias of it in the menu bar subject and pass the *alias* to Open Window. If, on the other hand, you copy the dialog Window object, paste the *copy* into the menu bar subject, and pass this to Open Window, then the menu item will open an exact *duplicate* of the dialog, but not the dialog you want.

An object can see only objects that reside in its subject. If, for example, a Menu Bar object is in one subject and its Menu objects in others, the Menu objects will not appear in the Menu Bar's editing dialog. As a result, you won't be able to add the Menu objects to the Menu Bar.

If, on the other hand, the Menus are aliased in the Menu Bar's subject, then the Menu Bar will recognize and display them in the editing dialog. You then add them to the Menu Bar just like ordinary Menu objects.

Before you can alias an object, you must *share* it. Done only once, sharing publishes the existence and whereabouts of the object throughout the project.

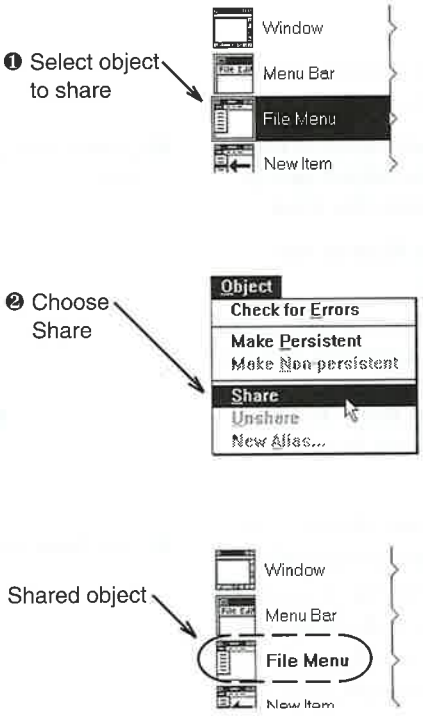
Once shared, an object can be aliased in any subject within your project. Shared objects appear in the object group and are distinguished from other objects by their boldface titles.



Sharing an Object

1. **Select the object(s) in the object group.**
2. **Choose Share in the Object menu.**

   Notice that the object title appears in bold letters. This indicates that the object has been shared.

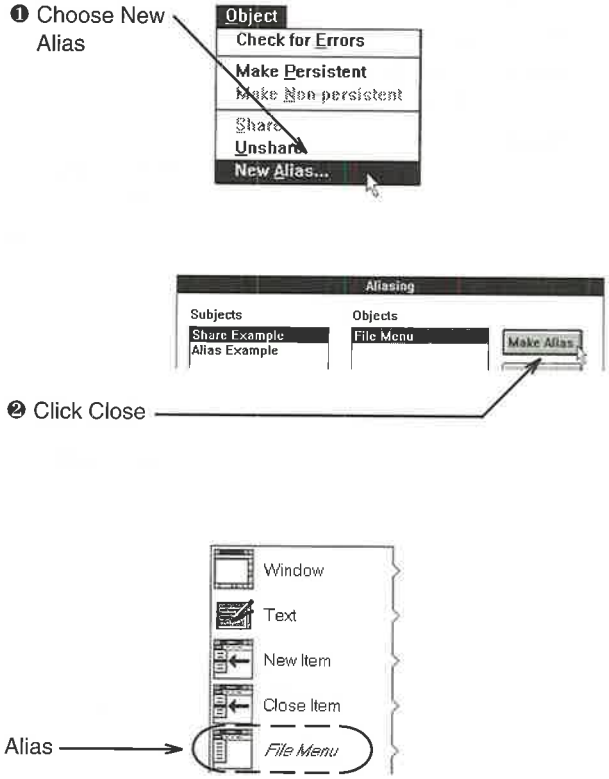


☛ **Note** An object can't have aliases if it is not a shared object.

Making an Object Alias

Now that the object has been shared, it can be aliased in other subjects.

1. **Open the destination subject and select the object group that will contain the alias.**
2. **Choose New Alias in the Object menu.**
3. **In the Subjects list, select the subject that owns the shared object(s) in the Objects list, then click the Make Alias button. Click the Close button.**



An alias of the shared object(s) now appears in the object group of the current subject. The title of the alias is italicized to help you distinguish the alias from objects actually resident in that subject. Nevertheless, there is nothing you can do with other objects in the list that can't also be done with the alias. It displays a signal list you can use to call function chains and you can connect it as a parameter to any function in its subject. It can also be included inside other objects (Database, Group, Menu, Window, etc.).

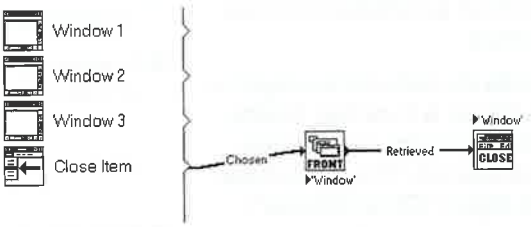
☛ **Note**

1. Objects cannot be shared between projects.
2. You cannot make an object alias a shared object, so you can't make an alias of an alias.

Aliasing Example

This example presupposes some familiarity with the Subroutine object. You may want to read about the Subroutine object in *Essentials ALMs*.

Let's suppose that you're building an application that has multiple windows. You've set up a File menu containing among other things a Close item. When you select this item at run time, you'd like your program to close the front-most window. A subject to do this might look like this.



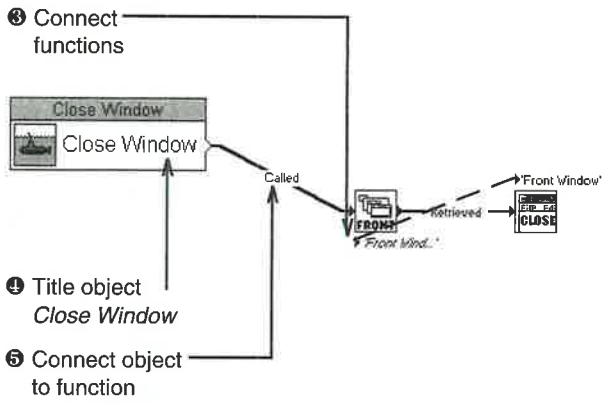
Here's how this function chain works. When you select the Close menu item, it issues a Chosen signal to call the Front Window function. This identifies the front window and passes it to Close Window, which closes the window.

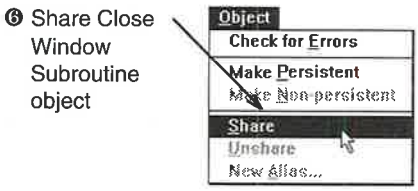
Now, suppose that your application uses the Front Window-Close Window routine often enough that you've decided to define it as a subroutine, and that you want to put this subroutine in another subject, together with the rest of your subroutines. Your problem is how to connect Close Item to that subroutine.

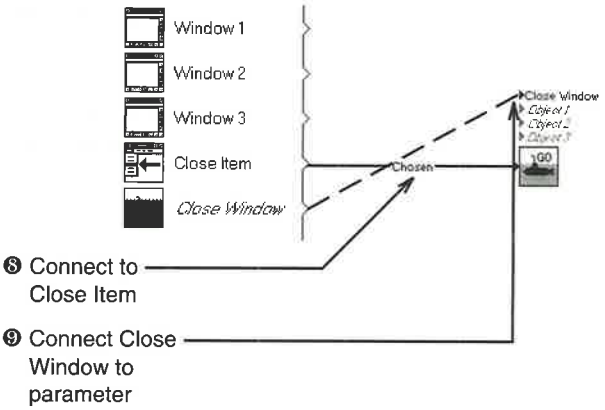
continued...

This is what you do:

1. **Cut the Front Window and Close Window functions from the subject (let's call this the Main subject).**
2. **Create a new subject called Subroutines, and paste in the two functions you just cut.**
3. **Link these two functions with a flow and connect the output parameter of Front Window to the input parameter of Close Window.**
4. **Locate the Subroutine object in the Object & Function Palette and drag it into your Subroutines subject. Title the Subroutine object "Close Window".**
5. **Draw a signal between the Subroutine object and the Front Window function.**
6. **Share the Close Window Subroutine object.**
7. **Bring the Main Subject to the front and select the Subroutine category in the Object & Function Palette to display the Subroutine functions.**
8. **Drag the Call Subroutine function into the Main subject and connect it to the Close Item object.**
9. **Alias the Close Window subroutine in the Main subject and draw a parameter line between the alias and the Call Subroutine function.**







Editing & Deleting Object Aliases

You cannot directly edit an alias, since it is only a surrogate for the original object. When you double-click an object alias, the subject containing the original object is brought to the front and the shared object is selected. To edit the object, double-click it.

You delete object aliases just as you do real objects.

1. **Select the alias to delete.**
2. **Press the Delete key or choose Delete from the Edit menu.**

☛ **Note** Deleting an object does NOT delete its aliases. See the next section.

Renaming, Disconnecting, & Reconnecting Object Aliases

Object aliases are uniquely linked to their original by type and title. The type is the category to which a given object belongs (Button, Database, Group, Window, etc.). The title is the name you assign to an instance. While you can't change a shared object's type, you can of course modify its title. If you do so, you are asked whether you want to rename the object's aliases as well. If you do, links between shared original and aliases are preserved. Otherwise all links connecting the object to its aliases are broken. However, the aliases remain. Once disconnected from their original, the aliases are like loose wires. You can reconnect them to their original object, link them to some other shared object of the same type and name, or remove them altogether.

One significant advantage of programming with object aliases is that they make it easy to replace your subjects or even use other people's subjects without a lot of editing. If you want to replace a subject in your project, just remove it using Cut or Delete. All alias links to other subjects in the project are immediately severed. Just as nice, however, is the way subjects can be reconnected. When you replace a subject, alias links to other subjects are automatically reestablished, provided that aliases can find matching shared originals in the new subject.

To illustrate how this works, let's consider one of the examples used earlier (Sharing and Aliasing Objects). Suppose that you're working on a project containing two subjects, one for a menu bar and its constituents, and another for your application windows. Several menu items in the menu bar subject are attached to Open Window functions whose input parameters are aliases of Window objects in the windows subject.

During the development process, you decide to replace the windows subject.

1. Replace the windows subject. Leave the Window object aliases in the menu bar subject intact.
2. Title the Window objects in your new windows subject just as you did those in the old subject.
3. Share the new window objects.

These new windows are different objects, but, as long as they are shared and their names match those of the aliases in the menu bar subject, the alias links will be automatically reestablished.

If you try to compile an application containing an alias for which there is no matching original, the AppWare environment alerts you to the absence of an original. However, it will allow the compilation from proceeding.

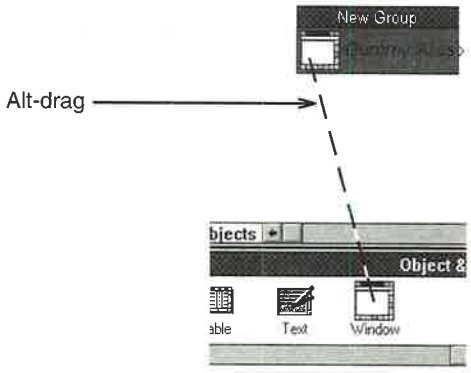
Creating Dummy Aliases

Creating an alias, as you've learned in the preceding sections, involves selecting and sharing an object in one subject and then aliasing it in another. This procedure works fine as long as you have access to the original. But, if you're working on a project that is being designed by several programmers, you may need to alias an object in someone else's subject, a subject that may not yet be available to you. How do you proceed?

You create a *dummy alias*, an alias that has as yet no connection to an original object. To create a dummy alias, do the following.

1. **Hold down the Alt key.**
2. **Drag an object of the type you want to alias from the Object & Function Palette.**

   This creates a dummy alias in the object group.



3. Title the alias with the name of the shared object that will be the original.

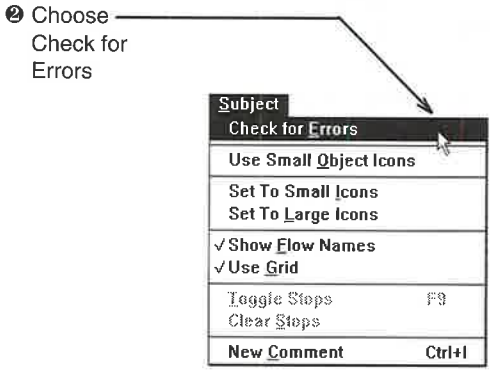
The dummy alias automatically links up with its original once the subjects containing alias and original are deposited together in a project, provided that the original has the same type and title as your alias and that the original is shared. To use dummy aliases, workgroup partners need only agree on title and sharing conventions.

Checking a Subject for Errors

Project error checking is automatically done at compile time. Before compilation, you can also check individual subjects for errors.

1. **Bring the subject window to the front.**
2. **Choose Check for Errors from the Subject menu.**

   Like the error checking that occurs at compilation time, this checks for programming mistakes such as missing parameters, aliases missing an original, etc. The program reports the nature of any errors it finds, but leaves you to correct the problems.



OBJECTS

The Object & Function Palette

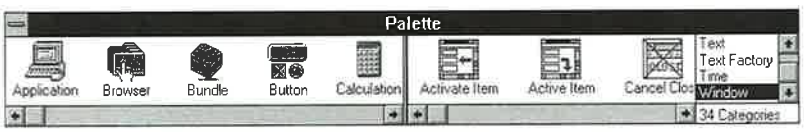
Objects are self-contained, reusable programming components containing both data and pre-compiled code needed for manipulating data. AppWare objects represent high-level application features that users will intuitively understand. The Window object, for example, enables your application to construct and operate windows. The Database object contains instructions for the operation of a database, and so on.

In AppWare, you get objects and their associated functions from the Object & Function Palette, described in the following section.

Object features and functions are described in the Essentials, and other ALM manuals.

The Object & Function Palette displays alphabetical lists of objects and functions available for use in AppWare. The objects displayed in the Palette are those located in the \CONFIG directory that's installed with AppWare. Functions in the Palette are owned by their respective objects.

See Copying Compiled Applications at the end of this chapter for further discussion of the Configuration directory.



The Palette has three scrollable compartments:

1. The left compartment displays objects. Only objects whose files are found in the Configuration directory are shown.
2. The right compartment displays a list of function categories. When you select a category, functions of that type appear in the center compartment.
3. The center compartment displays functions associated with the selected function category. These categories generally relate to an object type.

When you want to use an object of a particular type in your application, you drag its icon from the Object & Function Palette into the object group. AppWare then creates a copy or instance of that object type in the object group. Once an object has been copied into the object group, it can be named and edited.

When you Alt-drag an object to the object group, an object alias is created. See Creating Dummy Aliases.

Showing & Hiding the Object & Function Palette

The Object & Function Palette is a floating window that can't be covered up by other windows. You can, however, hide it. To hide the palette:

1. **Choose Hide Palette in the Windows menu.**

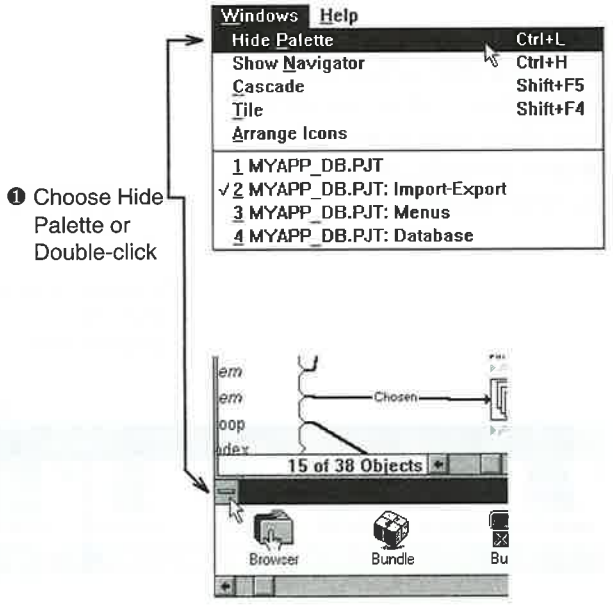
   or

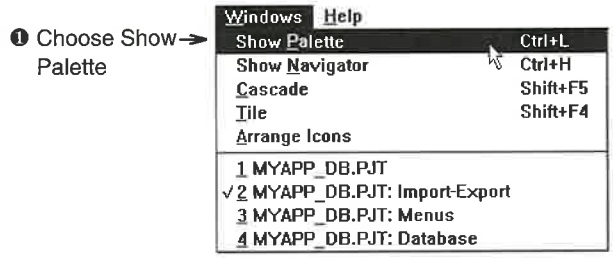
   Double-click the Palette's control-menu box.

Hiding the Palette is useful when you need a little extra work space for your function chains and you don't want to scroll the subject window. With the Palette hidden, you can make your subject window fill the screen. To do this drag its size box to the corner of the screen or click its Minimize/Maximize box.

To show the Object & Function Palette:

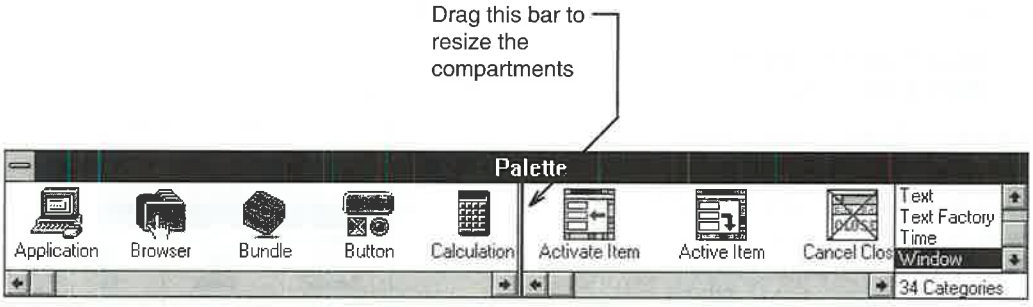
1. **Choose Show Palette in the Windows menu.**





Resizing the Object & Function Compartments

You move the vertical partition separating objects from functions by dragging it. Move the partition right if you want to see more objects, move the partition left to see more functions.



Choosing Function Categories

To add a function to the worksheet you first need to choose a Function Categories. The Object & Function Palette provides you with this feature.

To choose a function category:

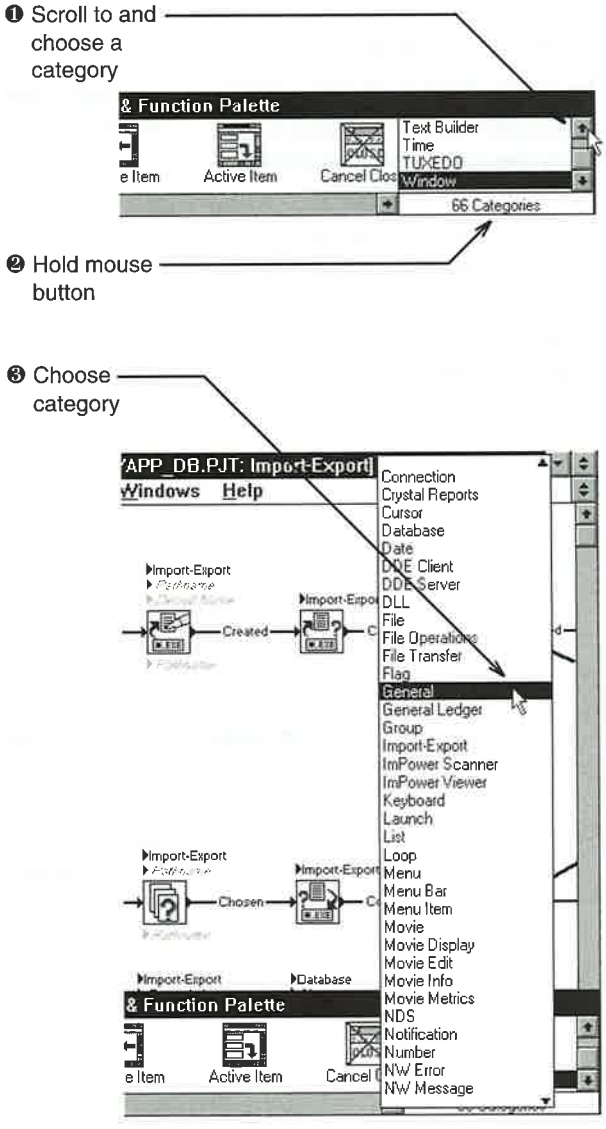
1. **Click on the scroll button in the Categories list and choose the desired category.**

   or

2. **Hold the mouse button on Categories.**

   A list of all the available function categories appears.

3. **Move the cursor up or down to scroll to the desired category.**



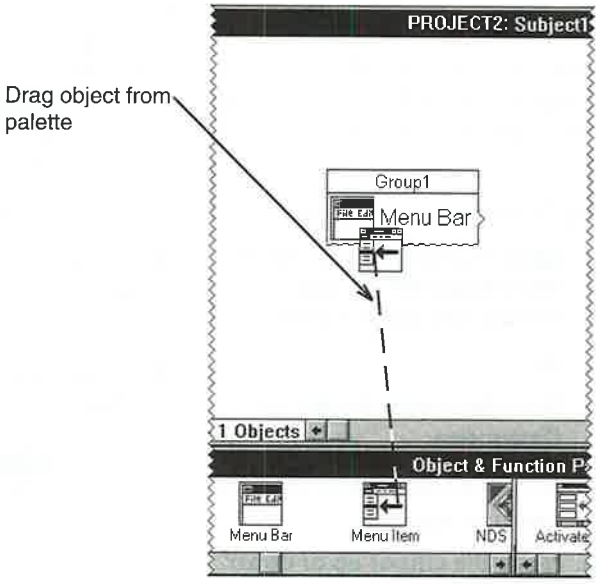
Adding Objects to an Object Group

The objects you use in a subject reside in groups. Every subject has one or more object groups.

To add an object to an object group:

1. **Double-click the object icon in the palette or drag it into the group as shown:**

   When you release the mouse button, the object is put in the highlighted object group. If no group is selected, a new one group is created.

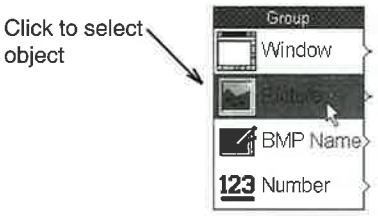


☛ **Note** Dragging an object into the object group creates a copy or instance of that object in the subject. Once in the object group, the object is given a generic title indicative of its type, such as Menu, Menu Item, Window, etc. You can retitle the object by selecting it and typing. See Retitling an Object below.

To the left of the title is an icon that identifies the object's type.

Selecting Objects in a Group

You select one object by clicking it. If you hold down the Ctrl key, you can select several individual objects. You can select a range of objects by clicking on the first object and shift-clicking on the last.



Deleting Objects from a Group

To delete an object:

1. **Select the object.**
2. **Choose Delete from the Edit menu or press the Delete key.**

   This removes the object and eliminates all of its flows and parameter links to functions. If the object is aliased, its deletion also leaves associated aliases without an original. Parameter links to the aliases are unaffected.

To delete multiple objects:

1. **Hold the Shift or control keys down and select the objects.**
2. **Choose Delete or press the Delete key.**

To undo a deletion:

1. **Choose Undo from the Edit menu.**

   Undo restores the object and all connections to it. AppWare supports a single level of undo.

To delete a group:

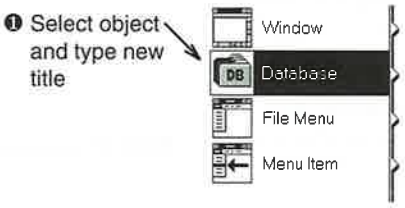
1. **Select the title bar of the group and choose Delete or press the Delete key.**

   This deletes all of the objects in the group.

Retitling an Object

When you drag an object to the object group, AppWare gives it a generic title, such as Menu Item for a Menu Item object. To retitle an object:

1. **Select the object and type new title.**



You may call your objects whatever you like, as their object group names have no effect on how your program will look and work at run time. Object titles need not be unique, except when you are aliasing objects.

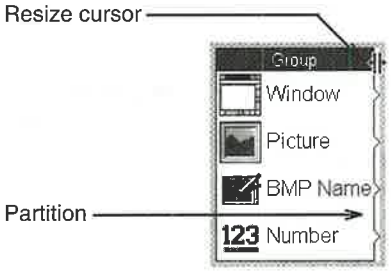
To edit an object title:

1. **Select the object.**
2. **Press the left or right arrow key.**
3. **Use the mouse or arrow keys to move the cursor within the object title and begin typing.**

Resizing Object Groups

The vertical partition on either side of the object group can be moved left or right. Click on the title bar of the object group, then drag the partition with the resize cursor.

Resizing the object group in this way is useful if you are working on a function chain and need a little more space in the worksheet.



Editing an Object

After adding an object to your project, you may need to edit it.

1. **Double-click the object or select it and press the Enter key.**

   A dialog box appears, allowing you to edit certain characteristics of the object. The look of this editing dialog varies according to object type. The various ALM manuals provide instructions for using these editing dialogs.

   If you're editing the object's title when you press the Enter key, the entire title is highlighted and the editing dialog does not appear. You must press Enter again to open the dialog.

   Alternatively, you can click the object icon and press Enter.

☛ **Note** You cannot edit an object alias. When you double-click an alias, AppWare opens the subject containing the original and then selects the original. As usual, you edit this by double-clicking it.

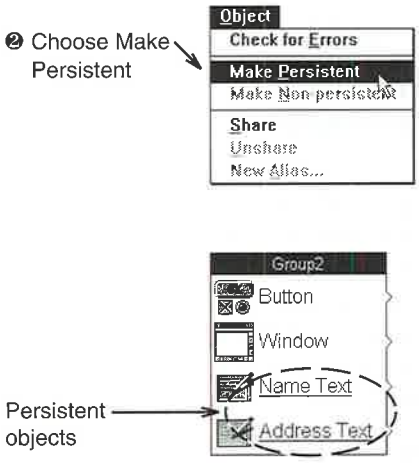
Persistent Objects

To make an object persistent:

1. **Select the object.**
2. **Choose Make Persistent from the Object menu.**

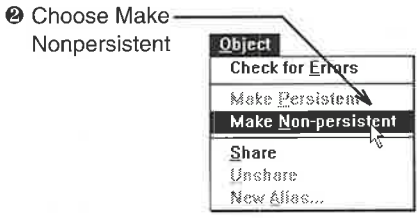
   The object now becomes persistent, and the object's title is underlined to indicate that the object is persistent.

Some objects cannot be persistent. Dialogs will notify you when you attempt to make such objects persistent.



To make persistent object nonpersistent:

1. **Select an object in an object group.**
2. **Choose Make Nonpersistent in the Object menu.**



Object Persistence

Persistence is the object attribute that enables an object to store run time data even after your application quits.

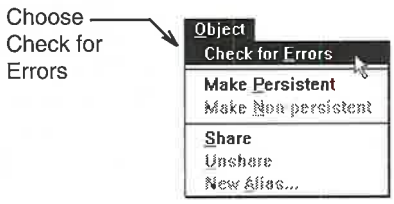
Without persistence, data entered in an object at run time is erased when you quit your application, unless it is saved in a file generated by an object such as Database or File capable of storing objects and creating files. When you launch your application, nonpersistent objects revert to their default values — the values contained in the objects when the project was compiled.

A persistent menu item, for example, stores whether it was enabled or disabled, toggled on or toggled off. When you relaunch your application, the menu item is in the same state it was when you quit. A nonpersistent item, on the other hand, reverts when you quit to whatever default setting you gave it when you compiled the application.

Checking an Object for Errors

1. **Select the object, then choose Check for Errors from the Objects menu.**

   This checks whether the object has all of the data it needs to operate.



☛ **Note** For example, say you were using a Calculation object to perform an arithmetic or algebraic operation on a few variable numbers. In this case, you must 1) specify a Number object to hold the result of the calculation, and 2) enter valid strings in the expression. If you did not correctly assign a Number object as an output variable in the Calculation object's editing dialog, an error would be detected when you chose Check For Errors. Likewise, if you did not use the correct variable name of a Number object in the expression (or made some other error in writing the expression), an error would occur.

When an error is found in your project, dialogs will appear describing what kind of errors have been identified.

Error checking is automatically carried out before you compile your application.

Using On-Line Help for Objects

You can display function information any time by switching to Help mode. Press Shift+F1 (the cursor changes to the Help mode cursor), then click an object icon in the Object & Function Palette or in the object group.

You can also click an object with the right mouse button to display the object's corresponding help screen.

FUNCTIONS

Using Functions

Before objects can be useful, the computer must know what you want your application to do with them. For instance, you may want to open or close a window, enter or delete a record in a database, etc. Each object comes with a set of functions that you use to control exactly how the object is to operate in your application.

The Window object's functions, for example, include Open Window and Close Window. You use these functions to open and close windows in your application. If your window contains editable text fields, then you might have occasion to use Text functions such as Append Return, Concatenate, Find-Replace, and so on. There are also File Operations and General function categories that comprise miscellaneous functions not associated with any specific object.

Unlike objects, functions contain no information, but can operate on information and interact with objects. An AppWare function performs a single, high-level operation. High-level means that the AppWare user is shielded from the complicated, operating-system-level programming that actually makes the function work.

Dragging a Function onto the Worksheet

1. **Select the desired function category from the list at the right end of the Object & Function Palette.**

   You can also select the category by clicking on the Categories label at the bottom of the list, and choosing an item from the pop-up that appears.

   Functions in the selected function category are displayed in the center compartment of the Object & Function Palette.

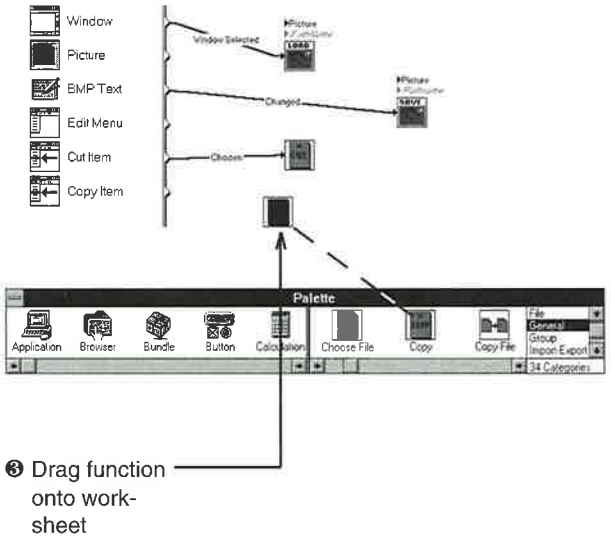
2. **Find the function you want by scrolling through the function compartment.**

   Functions are represented in the Palette by icons labeled and designed to illustrate what the functions do. The Functions are also listed in alphabetical order. For detailed information about how a given function works, consult the appropriate documentation or the on-line Help.

3. **Drag the function icon from the Palette onto the subject worksheet.**

   A copy of the function is created in the worksheet of the active subject. Put the function wherever you like.

continued . . .



☛ **Note** To delete a function from the worksheet, simply select it and press the Delete key or choose Delete from the Edit menu.

Duplicating a Function

You duplicate a function that is already on the worksheet by holding down the Ctrl key and dragging it. If you select several functions, then Ctrl-dragging duplicates all of them. The copies follow the movements of your mouse, and are left wherever you release the mouse button.

Function Chains & Signals

Objects and functions communicate with each other through events that are brokered by an event manager called the AppWare Bus. Most of the events used in an application are quite low-level and are therefore handled automatically. But as a user of AppWare, you are exposed to certain high-level events, called signals.

For example, the Menu Item object issues a signal when it is chosen, the Text object issues signals indicating when you change or edit its contents, activate it, deactivate it, and so on. You use these signals to call functions in response to actions by the user or occurrences generated under programmatic control. To indicate that an object calls a function, you draw a signal flow between the object and the function. Similarly, to indicate that one function calls another, you draw a flow between the functions. The order of the icons in the chain indicates the sequence of operations.

Most of the events that you see as signals are predefined by the ALM developer. A few objects can dynamically create new signals. In addition to its suite of predefined signals, the Window object, for example, issues signals to indicate when the cursor enters or leaves one of the Window items, such as a text field or a button. The Signal object even lets you define custom signals of your own. In this case, the object issues a signal when it recognizes a match between a "Selector" text and a text string you associate with that signal.

When you draw a signal connection from a function to an object or to another function, the connection is automatically labeled with the name of the default signal. In many cases, especially between functions, this may be the only signal. If other options are available, they are displayed in the pop-up menu that appears when you click on the signal name.

Occasionally, you may need to make several different connections between a pair of functions or between an object and a function. Suppose, for example, that you're using the Number function Compare Numbers to test two numbers. If the first is equal to or greater than the second, you want to display an alert. In this cases, you twice connect the Alert Notify function to Compare Numbers twice, once using the Equal signal, and then using the Greater Than signal.

The multiple signals generated by many functions are also the means by which you represent conditional statements in AppWare.

The following sections discuss these and other uses of the signal in more detail.

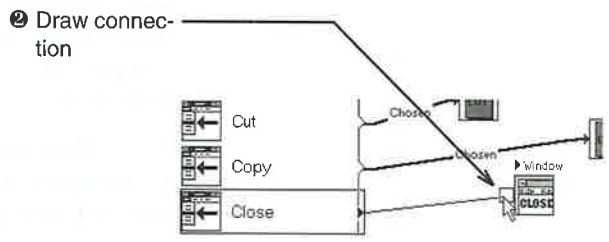
Connecting Functions to Objects

To create a signal flow:

1. **Click to the right of the object in the object group.**
2. **Draw a signal flow to the center left side of the function.**
3. **Release the mouse button.**

   Once you release the mouse button, a *connection* is made between the object and the function. The signal flow is labeled with the name of the signal that will trigger the function.

You can also draw the signal flow from the function icon to the object; the direction in which you draw the line has no bearing on the logic of the function chain.



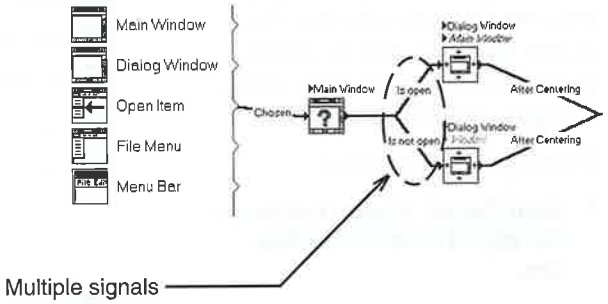
Connecting Functions to Functions

Links between functions work like object-function connections.

1. **Click to the right of the first function.**
2. **Drag a signal flow to the left side of the second function.**
3. **Release the mouse button.**

Functions, like objects, issue signals that you can use to call other functions. The flow between two functions, like the one between an object and a function, is labeled with a flow name.

Also like objects, functions may issue more than one signal. Through the use of its different flows, one function may call several different function chains that extend from the function like branches of a tree. Each of these branches represents a path along which your function chain may proceed.



☛ **Note** You may attach any number of function chains to an object or function using the same or different signals.

Editing Signal Connections

As noted in the previous section, each object and function has its own set of signals. Some objects and functions generate only one signal type; others generate several. This section shows you how to change a signal flow or select multiple signals to trigger a function chain.

To change an object signal:

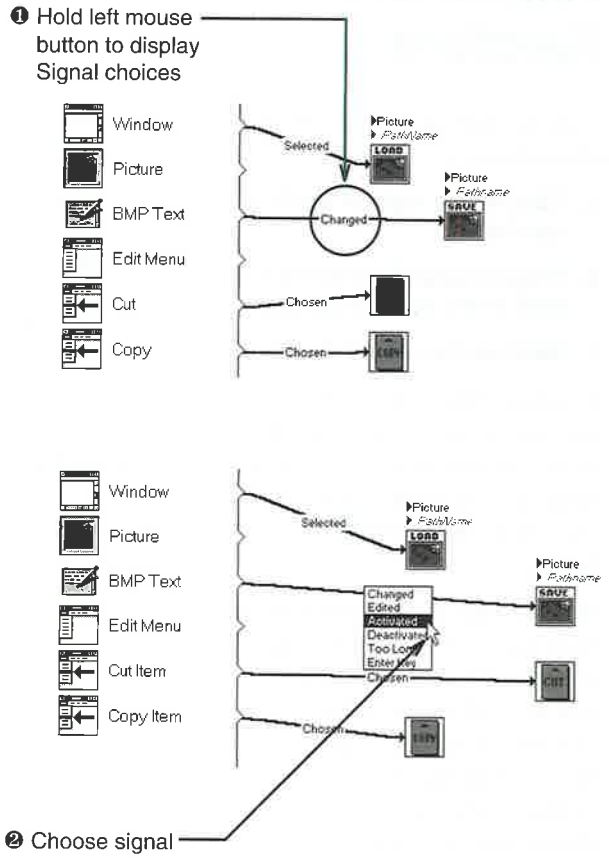
1. **Hold the left mouse button on the signal label of the flow line.**

   A pop-up menu appears, showing you all of the signals issued by that object as it is currently configured. (The pop-up's list of available signals changes depending on the object's settings. You'll learn more about this in just a bit).

2. **Choose the desired signal and release the mouse button.**

   The newly chosen signal appears in the signal flow label box.

Delete a signal connection by clicking on the signal name and pressing the Delete key or choosing Delete from the Edit menu.

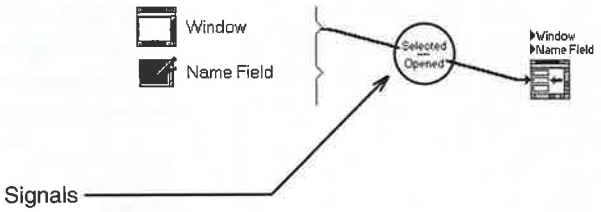


You can hide function signal names by unchecking Show Flow Names from the Subject menu. Numbers identifying the signals replace flow names in the label box. To redisplay flow names, choose (check) Show Flow Names.

An object can issue more than one signal to call a function. To select additional signals to trigger a function:

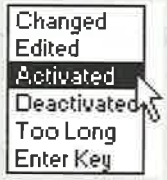
1. **Drag as many flow lines between the object and the function as you need.**
2. **For each flow, choose a different signal.**

   Selected signals appear one on top of the other in the signal flow label box. Each is individually editable and deletable.



Signal Choices

At a given moment, the selection of signals listed in the signal pop-up depends on the attributes of the object. For example, a normal menu item (an item in a pull-down menu) has only one signal: Chosen, issued when the menu item is chosen. A toggle menu item, on the other hand, can issue three signals: Chosen, Toggled On, and Toggled Off, respectively issued when the item is chosen, when it is switched to a toggled state, and when it reverts to its untoggled state.



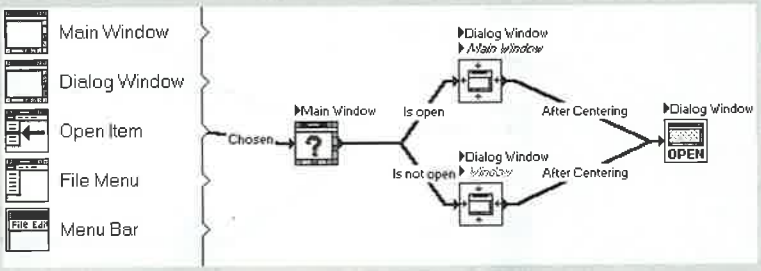
Another example: An editable Text object issues six signals: Changed, Edited, Activated, Deactivated, Too Long, and Enter Key. Except for Changed and Too Long, these signals are issued in response to user actions. If the object is uneditable, some of these signals cannot be generated. As a result, uneditable Text objects list only the Changed and Too Long signals in their signal pop-up.

As you look over an object's signal list, keep in mind that its makeup reflects the object's settings.

The multiple signals that many functions generate can be used to call different function chains depending upon the result of the function operation. It is through the use of such alternative signals that you build *conditions* into your function chain sequences.

Using Conditional Statements

Branches in a function chain represent different possible paths along which your program may flow. Which of the branches your program flow follows depends on a condition that you specify in advance. Consider, for example, the following function chain:



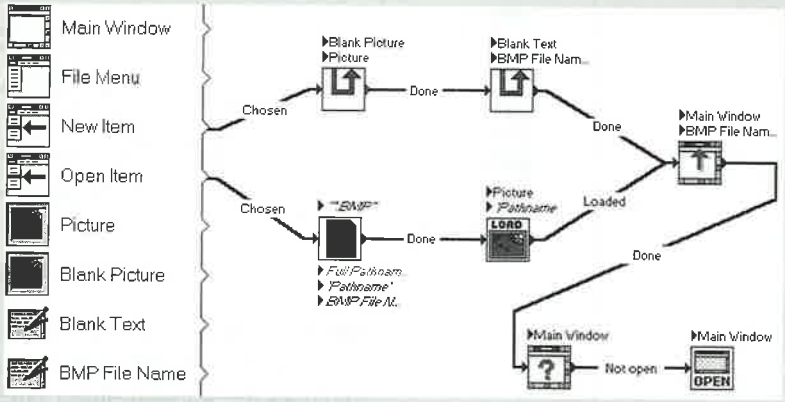
When the Open Item is chosen, it issues a signal to call the Is Window Open function. Is Window Open examines the Window object called Main Window to see if it is open. If it is, the function chain proceeds along the Is Open branch to the Center Window function, which centers Dialog Window over Main Window. After centering the dialog, the function chain continues on to Open Window, which opens the Dialog Window.

If Main Window is not open, the function chain proceeds along the Is not open branch, the Center Window function centers the dialog relative to the screen, and Open Window then opens the Dialog Window.

In conventional programming languages, this sort of branching is known as a condition. Many functions support such conditional structures. Consult *Essential ALMs* and other ALM manuals for more detailed information on how to use these functions.

continued...

A conditional statement is indicated when you see two or more function chains branching out of a single function. The inverse of this effect is the merging of two or more function chains into one, as seen in the following snapshot:



In this illustration, the function chains attached to the New and Open menu items merge at the end into a single chain that begins with the Window function, Set Title. The application represented here contains a window ("Main Window") that displays a picture. The New menu item clears the picture and the window's title bar and opens the window, if it is not already open. Open loads a .BMP file chosen by the user, assigns the name of the file to the window's title bar, and opens the window, if it is not already open. Let's look first at each of the separate chains and then at the sequence of functions they have in common.

The function chain executed by New Item begins with two Assign functions. The first of these clears the window picture by overwriting its contents with a blank picture. The next Assign copies the contents of an empty Text object into another Text object (BMP File Name) in which the current window title is stored. After the second assign, program control passes to the shared function sequence.

The function chain executed by Open Item first calls Choose File Name, which brings up a dialog allowing the user to choose the file to open. The function outputs both the path and file names of the chosen file. The path name is passed to the Load Picture function, which loads the .BMP file into the Picture object displayed in Main Window. The file name is passed to the .BMP File Name Text object. Program control then passes to the shared sequence.

The first function in the shared sequence sets the Main Window title using the contents of the Text object entitled .BMP File Name. Depending on which of the menu items was chosen, this object will either be empty or contain the file name of the newly opened .BMP file. The next function, Is Window Open, examines whether the Main Window is open. If it is, nothing happens; if it is not, then the Open Window function opens it.

A shared sequence of functions like the one illustrated here is somewhat like a subroutine, in that like a subroutine, a shared sequence limits code duplication by letting you use a single function chain more than once. However, shared sequences are not subroutines. The reason is that once you enter the shared function chain, there is no way to return independently to function chains that "call" it. In other words, unlike a true subroutine, a shared function sequence cannot be referenced as an entity in its own right.

The other disadvantages of shared sequences are 1) that they do not operate across subjects, since you cannot drag function lines between subject windows, and 2) that they tend to create situations that require flow lines to cross one another, which makes it more difficult to trace program flow.

For these reasons, when you have a sequence of functions that you use more than once in your application, it is usually advantageous to define it as a true subroutine rather than simply linking other function chains to it. To define a subroutine, use the Subroutine object. Documentation for this object is provided in *Essential ALMs*.

Input Parameters

Functions by definition operate on objects. The object or objects that a function does something to are its *parameters*. Parameters are of two types: inputs and outputs. The following sections discuss each of these in turn.

The labels above a function icon are the inputs to which you pass parameters. You pass an object as an input parameter to the function by connecting the object to one of the labels above the function icon.

To locate the object connected to an input parameter:

1. **Double-click the input label.**

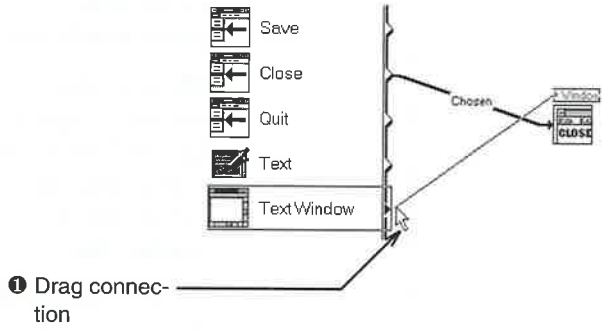
The object group scrolls to and selects the object.

To define an object as an input parameter:

1. **Click the label above the function and drag a line from the label to the object in the object group.**

   You can also drag the line from the object to the label or use the right mouse button to click on a function parameter to display a pop-up list of valid objects.

2. **Release the mouse button.**



☛ **Note** The line you drag from the label is visible as long as you hold the mouse button down. Once the tip of the line is within a few pixels of the object, a box appears around the object. This box helps to ensure that you don't inadvertently make a connection to the wrong object. When you release the mouse next to the object, the line disappears. At the same time, the label above the function is renamed with the object title, and the label text turns from gray to black. The change of label name indicates that a connection now exists between the object and the function.

When you use a function input or output parameter, you must have an object of that parameter type somewhere in your project. For example, you can't use the Date output of the Current Date function without having a Date object someplace in your project, even if you are simply passing that Date output directly to another function.

Output Parameters

Many functions generate data in the form of *output parameters* that can be passed to objects or to functions.

Passing an Output Parameter to an Object

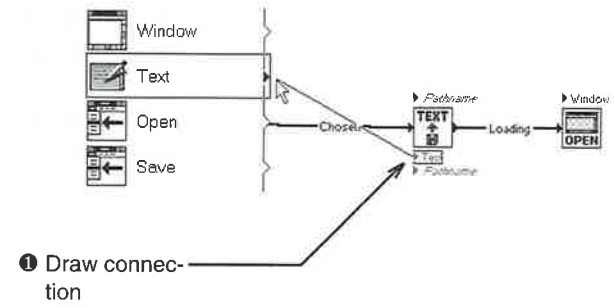
To pass an output parameter to an object:

1. **Click the label below the function and drag a line from the label to the object in the object group.**

   You can also drag the line from the object to the label or use the right mouse button to click on a function parameter to display a pop-up list of valid objects.

2. **Release the mouse button.**

A function's output parameter can be passed to only one object. If you want to pass a given parameter to multiple objects, you should use the Assign function.



☛ **Note** If a function's output parameter is connected to an object, it cannot also be passed directly to a function. Use the object previously assigned as an input for the function.

An object connected as an input parameter to a function can both pass information to the function and receive information from the function.

To locate the object connected to an output parameter, double-click the output label. The object group scrolls to and selects the object in question.

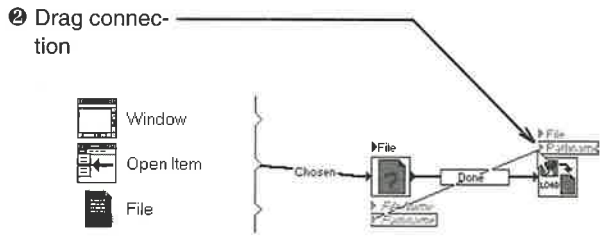
Passing a Parameter to Another Function

You can pass parameters from the output of one function to the input of as many other functions as you like. The source and destination functions need not be in the same function chain. However, when a parameter is passed between functions in different chains, the source function must have been executed before the destination function.

To pass a parameter between functions:

1. **Click the output parameter label below the source function.**
2. **Drag a line to the input parameter label above the destination function.**

   The line you drag is visible as long as you hold the mouse button down. Once the line reaches the destination label, a box appears around the label. This box appears only when the output parameter of the source is of a type that is compatible with the input of the destination.

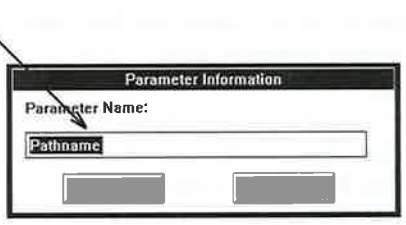


3. **Release the mouse button.**

   A dialog appears in which you can enter a name for this particular connection.

4. **Enter a name for this connection.**

5. **Click OK.**



The labels at either end of the line are renamed with the name you entered in the dialog bracketed by single quotation marks. Both label texts turn from gray to black. The change of name and the darkening of the labels indicate that a connection now exists between the two functions.

The single quotes remind you that this is a parameter connection between functions.

As with the signal flow between an object and a function, you can drag parameter links between functions in either direction.

The output of one function may be passed to as many function inputs as you like.

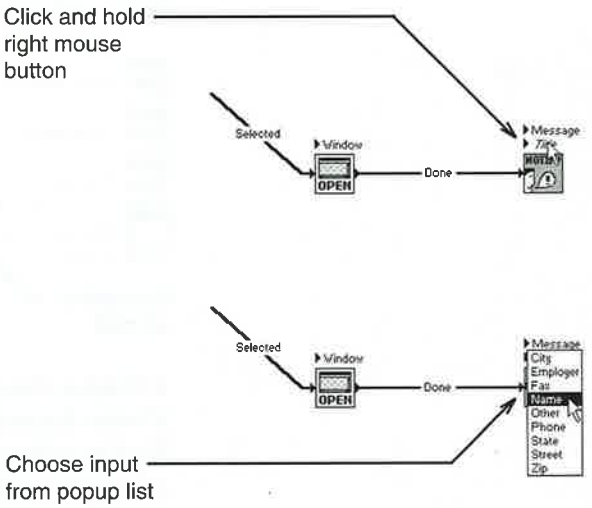
To identify the function(s) to which a given function output is connected, select the output label. The input labels of all connected functions are highlighted.

continued...

In some cases, you may not be permitted to connect one function's output to another's input. This is due to the fact that the input does not accept "pseudo-objects," temporary, dynamic objects used simply to pass data. In such cases, you must pass the output to an object, then pass the object to the other function's input.

Popup Parameter Lists

In addition to the line-drawing method of linking functions and parameters, you can use a shortcut feature called the Popup Parameter list. This feature simplifies the task of making connections between function parameters and objects. Both input and output parameters can be connected through popup parameter lists. The popup is activated by clicking with the right mouse button on a function on a function parameter.

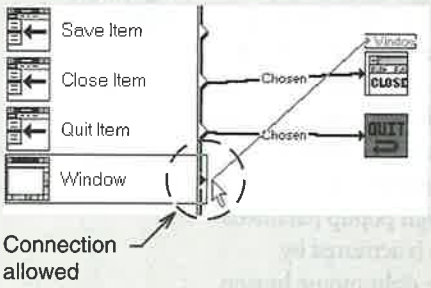


1. **Click and hold down the right mouse button on the parameter.**
2. **Choose desired parameter from the popup list.**

   The popup list contains objects within the subject that are of the appropriate type for the parameter you selected. Only objects within the current subject appear in the popup.

Parameter Type Checking

Input parameters and output parameters expect or produce certain types of information in the form of objects. Parameter *type checking* ensures that you connect only appropriate types of objects to function inputs and outputs. Consider this illustration:



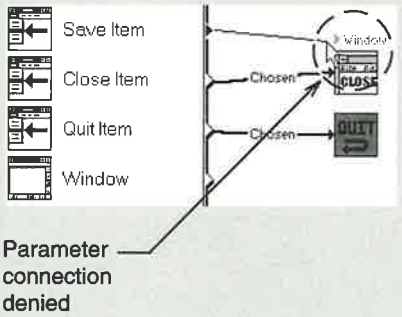
A Window object entitled Window is passed to the Close Window function. This connection instructs Close Window to close the Window. Since the application shown here could have any number of windows, Close Window must be explicitly told which window to close.

The required input to Close Window is a Window object. Passing a Text object to Close Window would confuse the function, since it can only operate on a Window. To prevent you from mistakenly connecting the wrong kind of object parameter to a function input, AppWare uses type checking. Type checking works by not allowing you to draw a line between a function label and an incorrect object type or between incompatible function outputs and inputs.

To see type checking at work, drag a line from a function label to an object group containing arbitrary objects. Hold the mouse button down and move the line up and down along the edge of the object group. You'll see that certain objects are framed and that their connection points are darkened while others are not. These two indicators show that a parameter connection is possible.

If a connection point remains white and no frame appears, then the objects in question are invalid parameters for that particular function input.

The same sort of type checking occurs when you drag the line from the object to the function. If the object is a valid input type for the function, a box appears around the function label. If not, no box appears, and no connection can be made.



Input parameters, Are like output parameters. Type checking prevents a connection from being made between a function output and an object of an incompatible type. For example, you can't pass the Date output of the Current Date function to a Window object. You can pass it to a Date or a Text object.

Type checking, limits the sort of parameters you can pass to and from a function. In some cases, type checking may limit the choice of object types to one. Many function inputs and outputs, however, do accept more than one type of object connection. For instance, certain functions have inputs and outputs that will accept any type of object. For these, there is effectively no type checking, and the input and output parameter types are therefore said to be global in nature. An example is the Call Subroutine function, which allows up to three optional parameters of any type to be passed to it.

continued...

Type checking also treats certain object types as generally interchangeable. These interchangeable types include Date, Number, Text, and Time objects. Interchangeability means that most Date, Number, Text, and Time functions will accept any of these four object types as parameters, even though Date functions generally operate on Date objects, Text functions on Text objects, and so on.

Declaring Constants as Parameters

The parameter that a function uses are of course objects. When passing information (especially variable information) to functions involving textual data, you typically use Date, Number, Text, or time objects. When the input is a constant (fixed text or number), however, you can declare it without the use of an object.

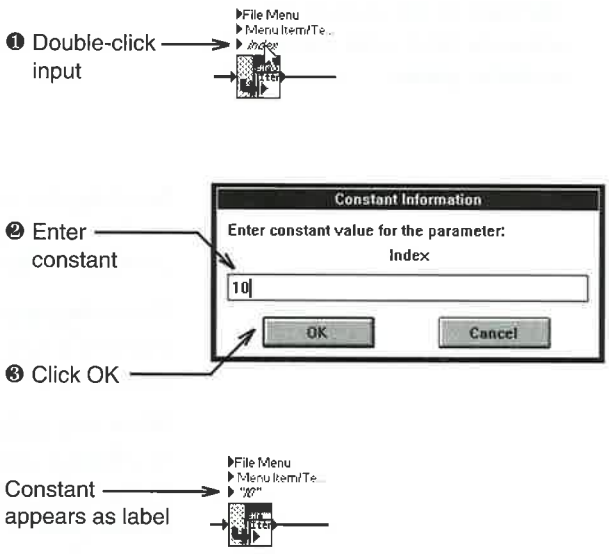
Suppose, for example, that you would like to set up a function chain to build dynamic menus — menus whose items are inserted and deleted at run time. You want the items inserted in the menu at a specific location, indicated by a fixed index number.

To implement this operation, you must pass the index number to the Insert Menu Item function. You can do this by entering the index in a Number object and passing that object to the *Index* input parameter of Insert Menu Item or you can enter a constant as the *Index* parameter.

To declare a constant as a parameter:

1. **Double-click the function's input label.**

continued...



A dialog box appears in which you may enter the constant.

2. **Enter the constant.**
3. **Click OK.**

   The value of the constant becomes the label name and appears in double quotes.

☛ **Note**

Anything that can be entered into a Text object can be used as a constant. Not all inputs allow you to enter constants in this way.

If no dialog appears when you double-click the label, then you'll have to input the parameter by means of an object.

When you declare a constant, you're instructing AppWare to create what is called a "pseudo-object". Unlike other objects, these have no visual representation in your project. They're temporary rather than permanent, and their values are fixed rather than variable. Many function outputs are also pseudo-objects;. For this reason, you may find from time to time that you cannot pass a function output to another function's input. The reason is that the input accepts only real objects. In cases such as these, you must first pass the output to a real object then pass that object to the other function's input. Inputs that accept pseudo-objects are noted in the ALM documentation.

Required & Optional Parameters

You'll notice that there are two styles for parameter labels: plain text and italic. The plain text style indicates that particular input or output is *required*. Italicized text means that the input or output is *optional*.

You must supply all required parameters before compiling your project or running it in the AppWare debugging environment. Prior to each of these processes, the environment checks all of the functions in the project to ensure that required parameters have been supplied. If any are missing, compilation will not proceed. The compiler returns you to your project and highlights the functions that lack their required parameters.

Some output parameters too may be required. While most are not, don't overlook those that are.

The Essentials ALMs manual and other ALM library manuals provide complete information about input and output parameters.

Moving a Function or Function Chain

1. **Select the desired function or function chain.**

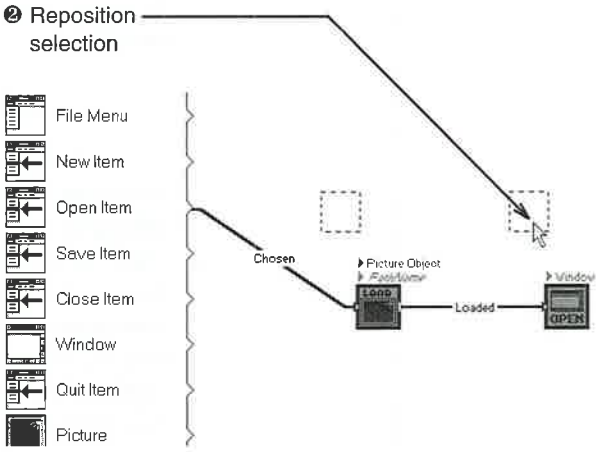
   Select a single function by clicking it once.

   Select multiple functions by holding down the Shift key and clicking them one by one. You may click also in an unoccupied area of the worksheet and drag a marquee around the functions.

2. **Drag the selection and reposition it with the mouse.**

3. **Release the mouse button.**

   Signal flows are redrawn to reflect the new function layout.



To select functions that are located off-screen:

1. **Click inside the worksheet near where you want the selection to begin and drag the marquee beyond the margin of the window until it encloses the functions.**

2. **Release the mouse button.**

   The worksheet auto-scrolls as you drag the marquee.

☛ **Note** The location of a function chain, by itself, does not affect the logic of your program.

Using On-Line Help for Functions

You can display function information at any time by switching to Help mode. Press Shift +F1, the cursor changes to the Help mode cursor, then click a function icon in the Object & Function Palette or in the subject worksheet. A function help screen appears, describing the function's input parameters, output parameters, and signals.

You can also click a function with the right mouse button to display the function's corresponding Help screen.

THE NAVIGATOR

Showing/Hiding the Navigator

The Navigator is a floating window that lists all open projects and their associated subjects. Subjects are listed under the projects that own them as are Comments.

The purpose of the Navigator is to give you quick access to any project, subject, or commented function chain in the AppWare environment.

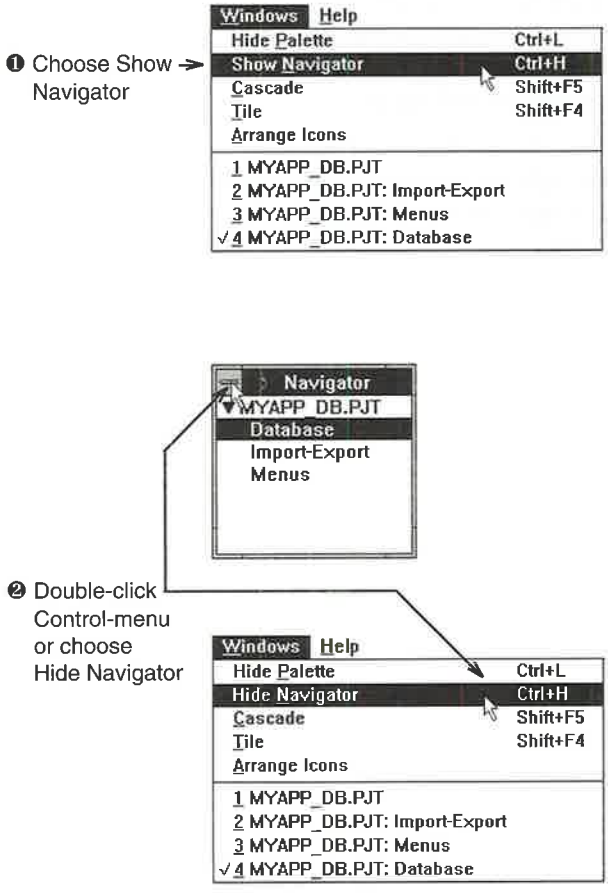
To display the Navigator:

1. **Choose Show Navigator from the Windows menu.**

As a floating window, the Navigator cannot be covered by another window. You may, however, hide it.

To hide the Navigator:

2. **Double-click the Navigator's Control-menu box or choose Hide Navigator from the Windows menu.**



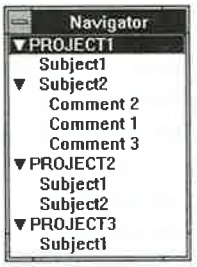
Using the Navigator

To the left of each project name in the Navigator is an arrow that allows you to display or hide the contents of the project. When the arrow points downward, the project's subjects are listed underneath the project name.

When the arrow points to the right, only the project name appears. You change the orientation of the arrow by double-clicking it.

Arrows also mark each subject, and can be used to display or hide comments in the subject.

Expanded list →



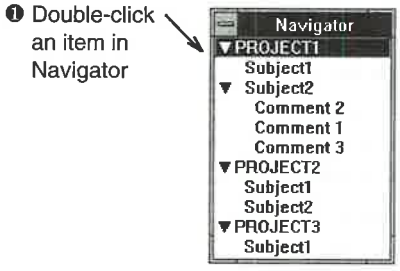
Condensed list →

To open a project or subject, or find a comment:

1. **Double-click the item name in the Navigator.**

For example, suppose that the particular project you are working on has several subjects. You know you have a certain function chain in your project, but you can't remember where it is located. You remember, though, that it was labeled with a comment, Save Data to File.

To find the Save Data to File function chain, open the Navigator window, look for the Save Data to File comment label and double-click it. The subject containing this comment is opened and its worksheet is centered around the Save Data to File comment. The comment briefly flashes to call attention to its location.



THE FIND DIALOG

Find Dialog Options

The Find dialog lets you locate items within a project. When a find is executed, the subject containing the first occurrence of the item is brought to the front and highlighted.

The Find What checkboxes let you limit the scope of the search. You must select at least one category.

The Range list lets you search the entire current project or restart the search to a specific subject. The default selection is the active subject.

The Search Options give you additional options for defining the scope of the search: Case Sensitive finds only those instances where the case of the item found is the same as the item you entered; Whole Word Match skips instances where the text is part of another word; Display Found List displays a floating list of subjects containing instances of the item you entered. All three search options are unchecked by default.

The Close button saves the current settings without initiating a search and closes the dialog.

Choose Find Next from the Edit menu or press the Ctrl+G to find the next occurrence of an item.

Finding Items in a Project

To find items in a project:

1. **Choose Find from the Edit menu or press Ctrl+F.**

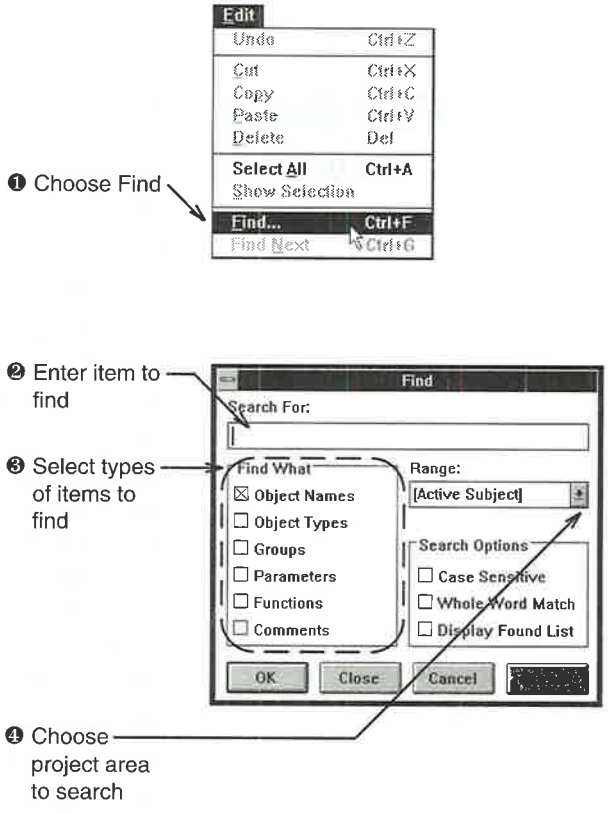
   The Find dialog appears.

2. **Enter the name of the item to find in the Search For edit box.**
3. **Select the type of item to find.**
4. **Choose an area to search from the Range dropdown list.**
5. **Select the desired options from the Search Options checkboxes.**
6. **Click OK to start the search.**

   The subject containing the first occurrence of the item is brought to the front and the item is highlighted.

   If you selected the Display Found List option, a list of subjects containing all instances of the item to find is displayed. Double-click a subject in the list to bring it to the front with the found item highlighted.

7. **Choose Find Next or press Ctrl+G to find the next occurrence of the item.**



DEBUGGING AND COMPILING

AppWare's Debugging Environment

After designing your application in AppWare, the final step is to debug and compile what you have designed.

Debugging involves running your application in an interpreter intrinsic to the AppWare programming environment. You can step through the application and find exactly where an error occurs. Knowing this, you can then address the problem and try again.

When you run in the interpreter, AppWare creates and launches a temporary application. When you exit the interpreter, the temporary application file is automatically deleted.

Debugging your application before you compile it is not necessary. However, you'll find it useful if you have a complicated application that doesn't work after it has been compiled.

Before running the interpreter, you should insert stops breakpoints along suspected problem function chains in your application. As the interpreter runs through your application, it pauses at each stop and waits for you to select Continue from the Project menu. If an error occurs between stops, you know that the problematic functions are to be found after the first stop and before the second. By inserting additional stops you can narrow down the problem even further. Although you can't edit an object, you may make other changes to your project while it is paused. These changes will not be reflected until you exit the interpreter and run the application again.

continued...

While your program is paused, you may double-click objects in object groups to examine their values. AppWare works in conjunction with the interpreter to display object values as your program executes from stop to stop.

Checking Project for Errors

At any time, you can check your project for errors. Error checking detects such things as aliases missing shared objects , unconnected parameters, or even mistakes within objects.

To check a project for errors:

1. **Select the window of the project to be checked.**
2. **Choose Check for Errors from the Project menu.**

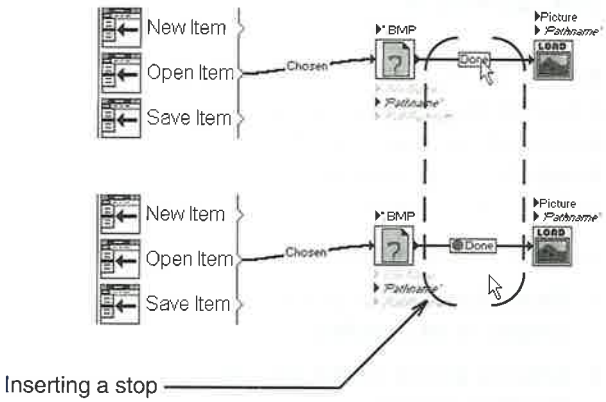
   AppWare will alert you if there is a problem and alert you as to the type of error found.

Inserting and Removing Stops

To insert a stop:

1. **Click the label of the signal or flow where you want to insert a stop.**
2. **Choose Toggle Stop from the Subject menu or press F9.**

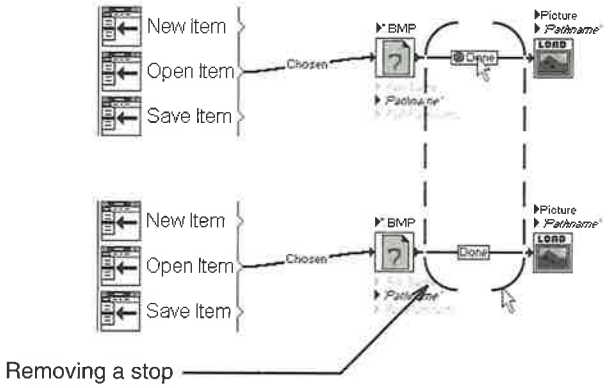
A stop is inserted on the signal flow. If you run your application using the debugger, your program will stop at this point in the function chain and return you to the AppWare environment. There you may double-click objects to examine their values. AppWare dynamically obtains these values from your application.



To remove a stop:

1. **Click the signal label with the existing stop you want to remove.**
2. **Choose Toggle Stops from the Subject menu or press F9.**

   The stop is removed.



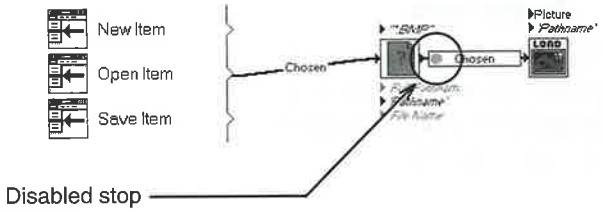
Enabling/ Disabling Stops

When you run your project in the debugger, it pauses at stops only if the stops Enabled item is checked. When unchecked, the stops, though still present and visible, are disabled.

Disabling stops is useful if you have previously placed stops in your function chains and you now want to try running the application without stops, but don't want to remove them yet.

To enable all stops:

1. **Choose (check) Enable Stops from the Project menu.**



Removing All Stops

1. **Choose Clear Stops from the Project menu.**

This removes all stops in your project. Do this once you've debugged your application and it's working as desired.

☛ **Note** You may leave stops in your compiled .EXE; they have no effect except when running in the debugging environment.

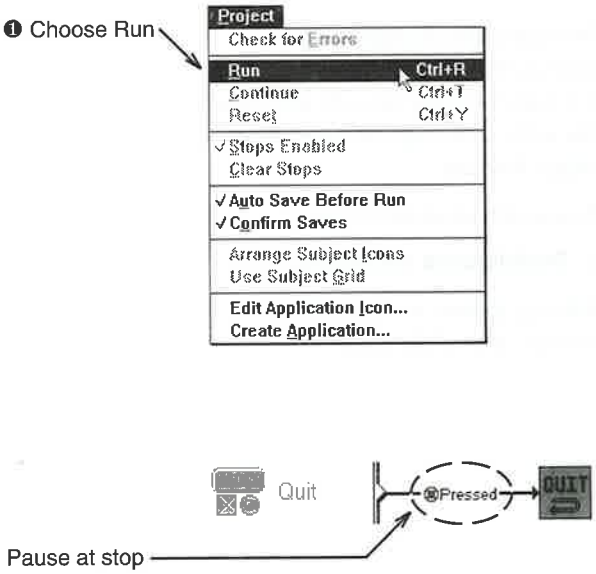
Running an Application Using the Debugger

1. Choose Run from the Project menu.

The application runs exactly as it would if it had been compiled using Create Application. It shows the same menus, dialog boxes, prompts, etc. that would appear if it were an executable application. The only difference is that it will execute a little slower than if it were compiled and running independently. Interpreters are very helpful in isolating problems, but they are costly in terms of overhead.

If Stops are enabled, the application pauses when it gets to a stop, and the stop icon is highlighted and briefly flashes to call your attention to its location.

If the stop reached is hidden from view, its subject is opened and scrolled to show the stop.



Examining Objects During a Pause

You may make changes to your project while your program is paused at a stop. You may not edit objects at this point. You may examine object values, however.

To examine an object value:

1. **Double-click the object.**

A dialog appears to display the current value of the object.

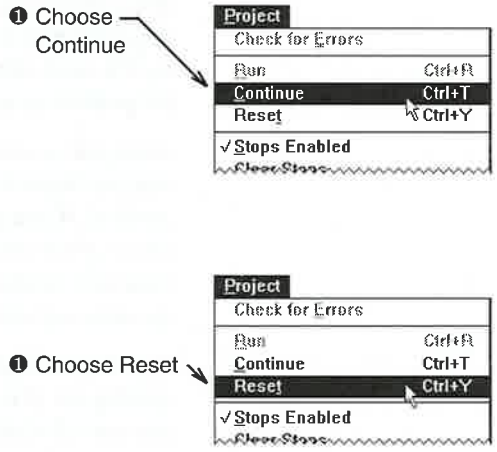
Continuing Execution After a Pause

To continue running your program after a pause:

1. **Choose Continue from the Project menu or press Ctrl+T.**

To abort execution after a pause:

1. **Choose Reset or press Ctrl+Y.**



Tips for Debugging

The first thing to remember to do before running the interpreter is to save your work. This can be automated by checking Auto Save Before Run in the Project menu. This saves all open projects before launching the interpreter.

When inserting stops, remember that it is unnecessary to pause before every function. Run the program first to see if it needs debugging. If it does, try to narrow down the problem to smaller and smaller areas of the program.

Once you've narrowed the problem down to a certain area, put stops around the functions you think are involved. If you put a stop both immediately before and after a certain function, for example, you can check the value of a number or text field before the function gets the value and after the function has modified it.

If you're looking at a function that has multiple flows coming out of it, put stops on all branches. This way you can tell which direction the program is flowing.

If the function you want to examine is at the end of a chain, but you want to examine a value after the function has executed, add a dummy function (an arrow cursor or something else that does nothing important) after the function you want to look at, and place a stop on the flow between the two functions.

Notice that the keyboard equivalents for the Run, Pause, and Continue options are Ctrl+R, Ctrl+T, and Ctrl+Y.

Memory Problems

Because both AppWare and your application are loaded into memory when you run the interpreter, you may occasionally run into memory limitations. There are a few things you can do to solve this:

- Exit other applications
- Use virtual memory
- Install additional RAM

COMPILING A PROJECT

Compiling

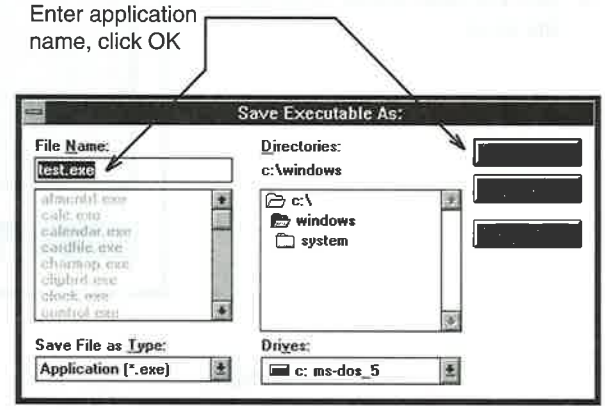
When you are finished designing and debugging a project, you can compile it into an application. All of your subjects, objects, functions, and function chains are merged into one program.

To compile a project:

1. **Bring the project or one of its subjects to the front.**
2. **Choose Create Application from the Project menu.**

   When you choose this menu item, a dialog box appears allowing you to enter a name for your application.
3. **After entering a name for your new application, click OK.**

   The compilation status window indicates AppWare's progress in creating your application.

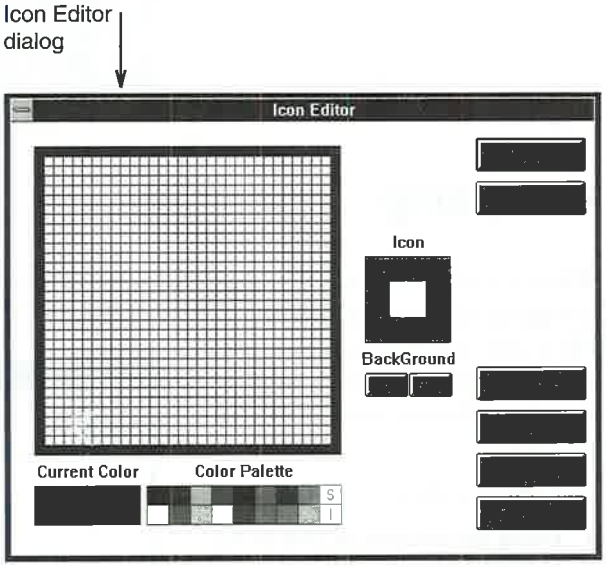


☛ Note If an error is encountered, it cancels the compilation process, informs you of the problem, and allows you to fix it. The error checking performed by the compiler is the same error checking that can be invoked manually by choosing Check for Errors from the Project, Subject, or Object menu.

Editing the Application Icon

To edit the application icon:

1. **Select the Project or Subject window.**
2. **Choose Edit Icon from the Project menu.**
3. **Use the icon editor to draw the icon.**



☛ **Note** The icon editor displays a "fat-bits" icon editing box. The view is enlarged to make your editing easier. Each square or bit in the fat-bits editor represents a single pixel in the icon. To draw the icon, select a color from the color palette, point and click or drag to draw lines.

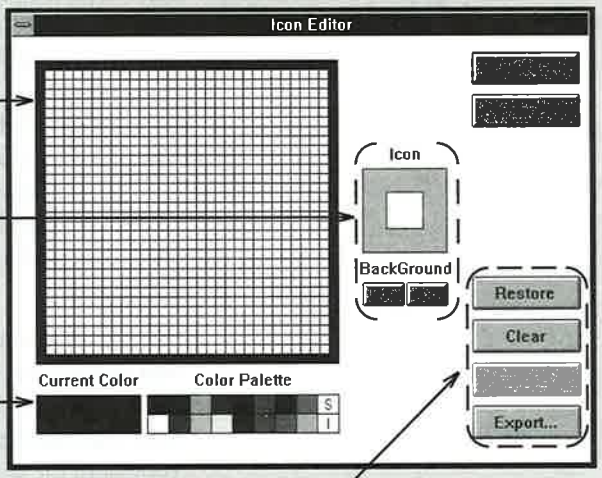
The Current Color box displays the current drawing color. The Color Palette provides 18 color options, including current screen's background color (s) and the inverse of the current screen's background (i).

An example of the icon appears in the Icon box at the bottom right of the dialog. The Background buttons allow you to scroll through the available background color examples.

The Import button allows you to import any icon (.ICO) file.

Icon Editor Dialog

The Icon Editor has the following options and features:



Drawing grid for creating/editing icons

Example of icon on background, buttons scroll through background colors

Current color and Color Palette

Restore, reverts to original icon

Clear, clears the drawing grid

Import, imports *.ICO files

Export, exports *.ICO files

Copying Compiled Applications

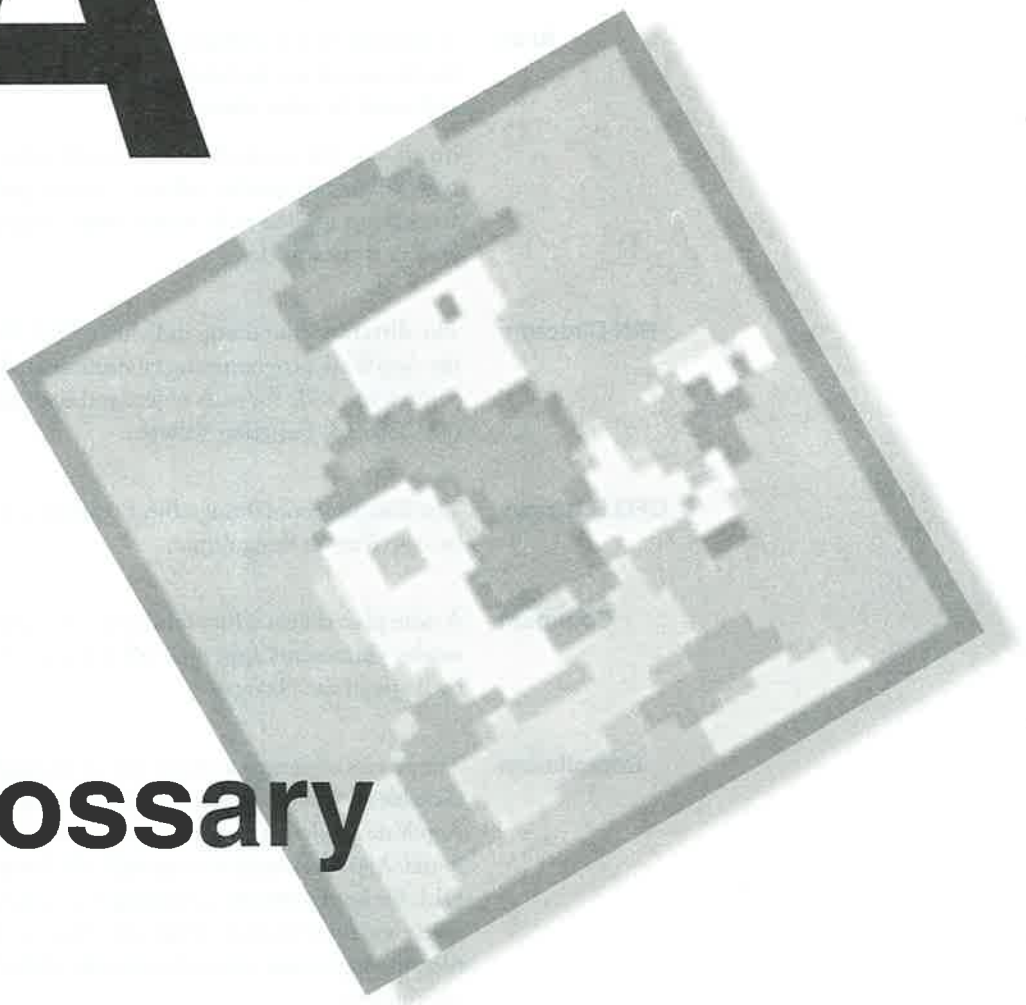
The resources that make your compiled applications work are not actually included inside the executable file. Rather, all the program resources are located in DLL files installed in a directory called \APPWARE\BIN. As new versions of the objects are released, simply replacing these files automatically upgrades applications you've already compiled. For example, if Novell releases a new version of the Database object that offers improved speed, in finding records, replacing the Database DLL enables compiled database applications to benefit from the higher performance object without having to be recompiled.

After you have built and compiled a new application, you may wish to move or copy it from time to time. Since your compiled application relies on the object DLL files installed on your computer, you will need to copy both the application and these files to the destination computer in order to run the program.

Upon installation, a directory called \APPWARE is created in the root directory. Inside the AppWare directory, two subdirectories, \BIN and \CONFIG, are created. The first is for AppWare and the ALM (Application Loadable Module) files containing the code that makes objects operate at run time. These files have a .DLL extension. There is generally one DLL file per type of object (ALM). The second directory contains CFG or configuration files which hold the object and function icons, parameter information, and other settings which are used only by the AppWare environment at design time. These files have a .CFG extension and are not accessed by the built applications at run time.

When you compile an application, AppWare creates a text file listing all of the ALMs required by the application. Identified by the extension ".REQ", this file has the same name as your executable and is located in the same directory.

A



Glossary

APPWARE TERMS

Alias A reference to a real object. Like a real object, an alias can be passed as a parameter to functions, issue signals, and reside in other objects.

An alias, is not a true object. It merely represents a *shared* object in another subject. Actions performed on an alias are applied to its shared object original when the project is compiled.

BIN Directory The directory containing individual DLL files for use in the AppWare programming environment. These files contain the code for each object and each function in the Object & Function Palette.

CFG Directory The directory containing object configuration files used by AppWare at design time.

Comment A note placed near a function chain to explain how it works. Comments appear as balloon icons and are listed by name in the Navigator.

Compilation The process of turning a project into an executable, double-clickable application. Before compilation proper begins, AppWare checks the project for syntactic errors. If none are found, AppWare builds an executable file that references a DLL file (see Extension) containing the requisite compiled object and function code. While dependent on these .DLL files, the application runs independently of the AppWare environment.

Compile Time The process of compilation. Compile time is preceded by design time and followed by run time.

Connection A parameter or signal flow link between a function and an object or another function. Parameter connections appear as black labels above or below function icons. A parameter connection is made when a function output or an object is linked to a function input or output label. Signal flows appear as dark lines linking functions to objects or to other functions.

Debugging Stop See Stop.

Design Time The period of time during which a project is designed. Design time is followed by compile time and run time.

Environment The AppWare tool used to design and compile a project. After compilation, applications are independent of the environment.

Extension A three-character addition to the standard eight-character file name. The three characters follow the period (.), as in the following example:

AUTOEXEC.BAT

Flow A signal connection between an object and a function or between a function and another function indicating one or more operations. Also referred to as a signal flow, or signal link. Represented on screen by a heavy line, each flow is uniquely linked to a signal which specifies the type of event that will invoke the function at the receiving end of the flow.

Function An operation performed by or on an object. Functions control the processes and logic of an application. Functions are accessible through the Object & Function Palette.

Function Chain A sequence of functions attached together by one or more connections.

Function Icon The representation of a function in the worksheet window.

Input Parameter See Parameter.

Instance A copy of a given object type used in a project.

Link See Connection.

Loop A repeating set of functions. A loop is constituted by a circle of functions. Loops can also be created through the use of the Loop object.

Object A programming module that contains data and the code needed to manage that data. Objects are the "nouns" — windows, menus, text fields, scanners, etc. — that perform an action or that are acted upon.

Object Alias See Alias.

Output See Parameter.

Parameter The connection made between a function and an object or another function that passes or retrieves data. Parameters are represented by labels above and below function icons. Parameters are different than flows in that flows control the order of operations in an application, while parameters control the flow of data.

Parameter Link The relationship between a function's parameter and an object or another function's parameter.

Persistence The attribute that allows an object to store its run time data. AppWare represents persistent objects by underling of the object title in the object group.

Precompile (Precompile Time) A synonym for Design time. See Design Time.

Program Flow See Flow.

Project The file that represents a program during design time. Projects contain individual subprojects, called subjects, each displayed in a separate window. Subjects in turn contain objects and functions.

Run Time Use of an application in the AppWare interpreter or following its compilation.

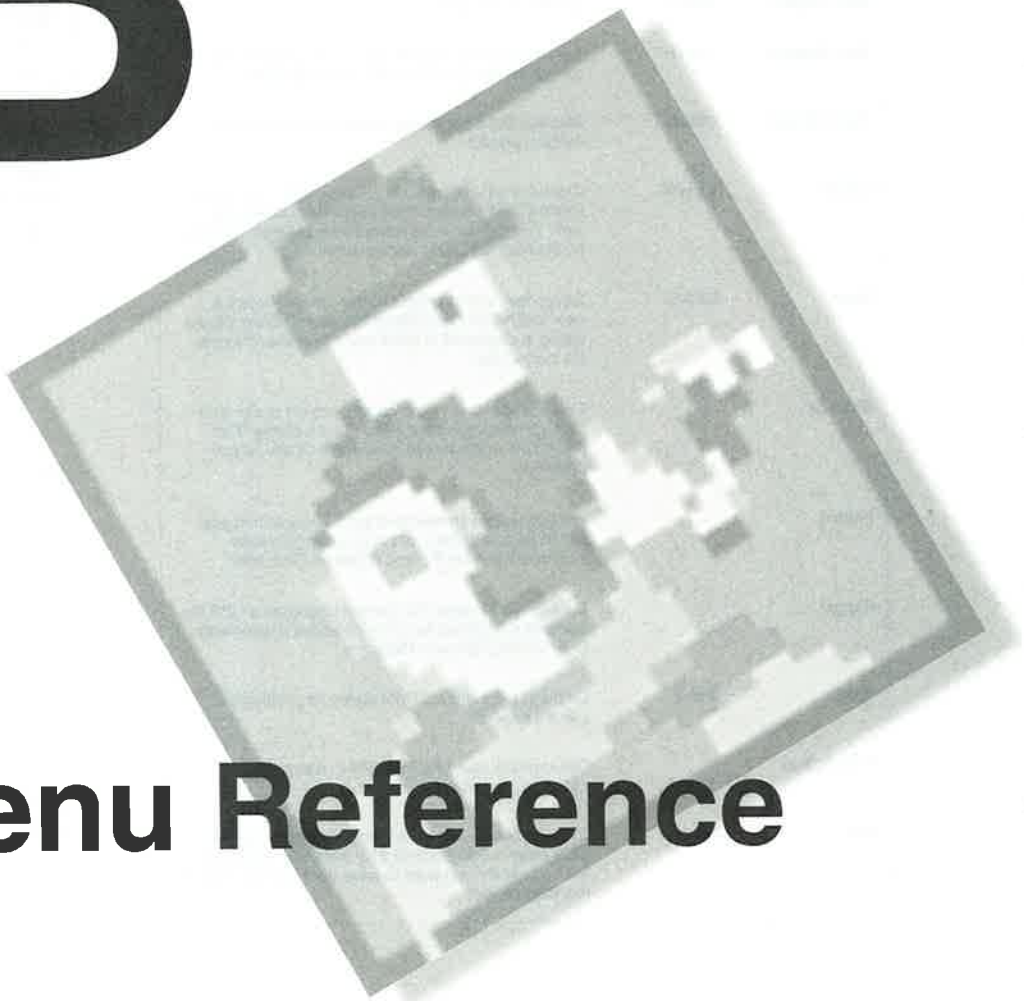
Shared Object The parent object of one or more object aliases. An object can't have aliases if it is not a shared object.

Signal Flow See Flow.

Stop A marker that instructs the AppWare interpreter to temporarily halt the execution of a test application running in the debugging environment.

Subject A subproject. Represented by an icon in the project window, each subject has its own window (subject worksheet) containing object groups and functions. Program logic resides in the subject.

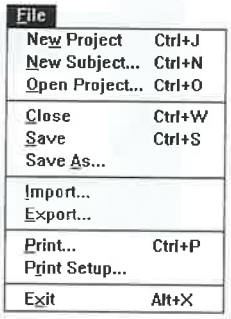
B



Menu Reference

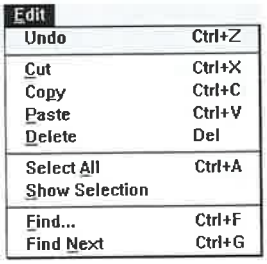
File Menu

Command	Hot Key	Description
New Project	Ctrl+J	Creates a new project.
New Subject	Ctrl+N	Creates a new subject in the current project. You cannot create a subject if no project is open.
Open Project	Ctrl+O	Displays the Open dialog allowing you to open an existing project.
Close	Ctrl+W	Closes whichever window is front-most. If the project window is front-most, the project file is closed as well. You can also close project and subject windows by double-clicking the control-menu box.
Save	Ctrl+S	Saves the current project. If the current project is new and hasn't been saved before, the Save File As dialog is displayed to allow you to name and locate the project file.
Save As		Enables you to save the current project under a new name by displaying the Save File As dialog. The project file under its previous name (if any) is not affected.
Import		Enables you to convert and load a project that was saved as a UPS file created in a non-Windows environment. See "Porting Between Platforms."
Export		Enables you to save the current project as a UPS file and port the project to a non-Windows environment. See "Porting Between Platforms."
Print	Ctrl+P	Displays the standard Print dialog for printing the current project.
Print Setup		Displays the standard Print Setup dialog for modifying printer settings.
Exit	Alt+X	Exits AppWare. If there are any open projects that have been modified since they were last saved, you will be asked if you want to save the changes to each modified project.



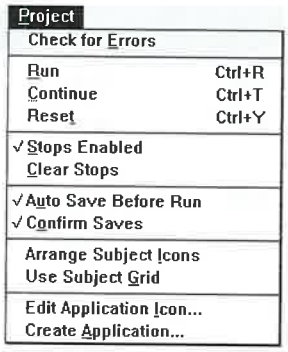
Edit Menu

Command	Hot Key	Description
Undo	**Ctrl+Z**	Reverses the last operation.
Cut	**Ctrl+X**	Deletes the selected items and copies them to the AppBuilder clipboard.
Copy	**Ctrl+C**	Copies the selected items to the AppWare clipboard.
Paste	**Ctrl+V**	Copies the contents of the AppWare clipboard to the front project or subject. If there is a subject on the clipboard, it is copied to the front project. If there are objects, functions, or function chains on the clipboard, they are copied to the front subject.
Delete	**Del**	Deletes the selected items.
Select All	**Ctrl+A**	Selects everything in the front project or subject. The title of this menu item changes to identify the items it will select, based on the identity of the front-most window.
Show Selection		Centers window around selected items.
Find...	**Ctrl+F**	Displays the Find dialog allowing you to locate items within a project.
Find Next	**Ctrl+G**	Finds the next occurence of the item specified in the Find dialog.



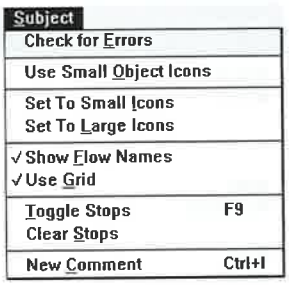
Project Menu

Command	Hot Key	Description
Check for Errors		Checks the current project for errors and indicates required solutions. Error checking is automatically invoked prior to running a test application in the interpreter or compiling a project.
Run	**Ctrl+R**	Compiles and launches a temporary test application based on the current project. If you have installed debugging stops and these are enabled, the test application pauses at each stop to allow you to examine values contained in its objects. Objects in the application communicate their values to their counterparts in the project.
Continue	**Ctrl+T**	Instructs the AppWare interpreter to resume running the test application after it has been paused.
Reset	**Ctrl+Y**	Instructs the AppWare interpreter to quit running the project after it has been paused.
Stops Enabled		Alerts the AppWare compiler to install the indicated debugging stops when the project is run in the interpreter (see Run above). If this item is not checked, the compiler ignores all stops.
Clear Stops		Removes all stops in the current project.
Auto Save Before Run		Alerts AppWare to automatically save all open projects prior to running or compiling a project. This protects open projects in the event of an unexpected, fatal error.
Confirm Saves		Works in conjunction with Auto Save Before Run (described above) by allowing you to confirm or cancel saving a given project.
Arrange Subject Icons		Arranges subject icons in the project window.
Use Subject Grid		Installs an invisible grid for aligning subject icons. Icons you subsequently move snap to the nearest grid lines.
Edit Application Icon...		Displays the Edit Icon dialog to allow you to create a unique icon for your application.
Create Application...		Displays the Save Executable As dialog in which you enter a name and location for the compiled project. When you click OK, the AppBuilder compiler builds a compiled application from the current project.



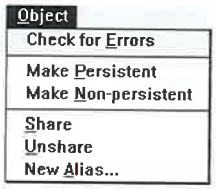
Subject Menu

Command	Hot Key	Description
Check for Errors		Checks the current subject for errors. Error checking is automatically invoked prior to running or compiling a project.
Use Small Object Icons		When checked, all subsequently added objects are displayed as small (16 x 16 pixel) icons. When unchecked, they are displayed as large (32 x 32 pixel) icons (Default).
Set To Small Icons		Displays small (16 x 16 pixel) object icons to the left of the object titles in selected object groups.
Set To Large Icons		Displays large (32 x 32 pixel) object icons to the left of the object titles in selected object groups.
Show Flow Names		When checked, displays the names of signal flows in the current subject (Default). When unchecked, replaces signal flow names with their index numbers.
Use Grid		When checked, installs an invisible grid for aligning groups and function icons. Functions subsequently placed on the worksheet or moved with the mouse snap to the nearest available grid lines. This is a cosmetic effect only. When unchecked, functions are located arbitrarily. Functions located on the worksheet, when this menu item is checked, do not reposition themselves unless they are moved.
Toggle Stops	**F9**	Inserts and/or removes debugging stops in the selected function signal(s). Stops in selected signals are removed. Selected signals having no stop receive one.
Clear Stops		Removes all stops in the current subject.
New Comment	**Ctrl+I**	Displays the Comment dialog allowing you to add a comment. In this dialog, you specify a brief comment (the comment name) and a full comment (the comment text). When you click OK, a comment icon labeled with the text of the brief comment appears in the center of the current subject's worksheet.



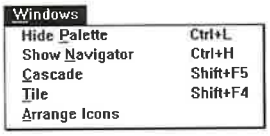
Object Menu

Command	Hot Key	Description
Check for Errors		Checks the selected object for errors and indicates the required solution. Error checking is automatically invoked prior to compiling a project or running it in the interpreter.
Make Persistent		Makes the selected object persistent. For more information about persistence, see Chapter 3.
Make Non-persistent		Makes the selected persistent object non-persistent. Also removes the underline of the object's name in the object group. For more information about persistence, see Chapter 3.
Share		Enables the current subject to "Share" the selected object with objects and functions in other subjects. For more information about sharing objects, see Chapter 3.
Unshare		Disables the sharing of the selected object. For more information about sharing objects, see Chapter 3.
New Alias...		Displays the Aliasing dialog allowing you to create an alias of a shared object in the current subject. For more information about aliasing, see Chapter 3.



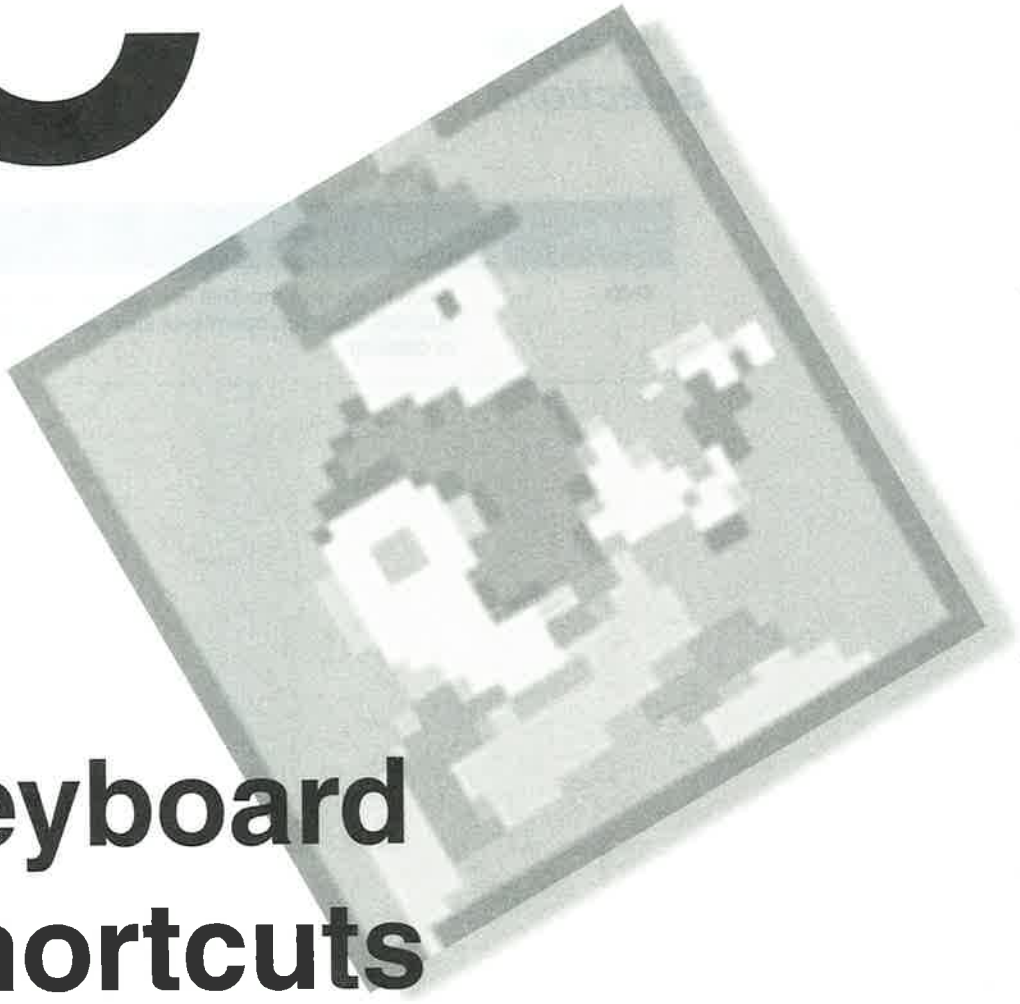
Windows Menu

Command	Hot Key	Description
Hide/Show Palette	**Ctrl+L**	Shows or hides the Object & Function Palette.
Hide/Show Navigator	**Ctrl+H**	Shows or hides the Navigator window.
Cascade	**Shift+F5**	Causes windows in AppWare to overlap so that each title bar is visible.
Tile	**Shift+F4**	Arranges open windows in smaller sizes to fit next to each other in the AppWare work space.
Arrange Icons		Arranges minimized Project icons in the AppWare window similar to the way minimized application icons are arranged by the Windows Program Manager's Task List.



☛ **Note** A list of open projects and subjects appears below the Arrange Icons menu item. To open a project or subject window, choose its name in this list.

C



Keyboard Shortcuts

PROJECT WINDOW SHORTCUTS

Multiple Selections

Hot Key	Action
Shift	If you hold down the Shift key, you may select multiple subjects for group operations such as copying, cutting, or deleting.

OBJECT GROUP SHORTCUTS

Selecting Multiple Objects

Hot Key	Action
Ctrl or Shift	You select multiple objects in an object group by holding down the Ctrl key and clicking the desired objects. You may also deselect objects in this fashion. You may select a range of objects by clicking the first object in the range, then Shift clicking the last object in the range.

Creating a Dummy Alias

Hot Key	Action
Alt	You create a dummy alias of a particular type of object by holding down the Alt key while dragging its icon from the Object & Function Palette to the object group.

SUBJECT WORKSHEET SHORTCUTS

Copying Objects/ Functions

Hot Key	Action
Ctrl + Drag	You can copy objects and functions on the worksheet by holding down the Ctrl key and drag the object or function to create a copy. This shortcut can be used to copy multiple objects and functions.

Selecting Multiple Functions

Hot Key	Action
Shift	You may select multiple functions and comments by holding down the Shift key and clicking the desired items. You may deselect an item in a group of selections by holding down the Shift key and clicking the item. You can also select multiple items by dragging a marquee around them.

Hot Key	Action
Ctrl + Alt	You can quickly select object groups and their entire function chains on the worksheet by holding down the Ctrl and Alt keys and clicking on an object group. You can also select segments in a function chain by holding down the Ctrl and Alt keys and clicking on a function. The function and those functions to the right are selected.

INDEX

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

W

NOVELL®

Novell, Inc.
122 East 1700 South.
Provo, Utah 84606
801-429-7000
801-453-1267

100-002638-001