Utilities and Developer's Notes

ACIUS

4th DIMENSION™

Acius

4th Dimension™ Utilities and Developer's Notes

4th Dimension and 4D Tools
by Laurent Ribardière

4D Renamer, 4D Customizer,
and 4D Mover by Ken Laws

This manual was written by Jeff Walden and Samir Arora.

Cover design by Patrick Chédal C & C.

Contents

Figures and tables

Preface

This supplement describes the 4th Dimension™ utilities supplied with your database:

- how to use the 4D Tools utility to clone an existing database (that is, to reproduce it *without* data), repair file damage, and resize database files (for multi-user systems only)
- how to use the 4D Renamer utility to automate the file-renaming process to change the name of an existing database

The supplement also covers topics of special interest to the 4th Dimension application developer:

- how to use the 4D Customizer utility to change a dozen apparently "built-in" attributes of the 4th Dimension program (or runtime version)
- how to use the 4D Mover utility to add externally programmed procedures to the 4th Dimension program (or runtime version) or to a particular database application
- how to operate 4th Dimension in a multi-user (networked) environment and set up a multi-user application

4D Tools utility

The 4D Tools utility for 4th Dimension clones databases, repairs damaged databases, and—as a required utility for multi-user systems only—resizes the record pointer allocation for a multi-user database.

Starting and quitting

4D Tools is a single program with its own icon. Start the 4D Tools utility from the Finder.

To start 4D Tools:

1. Double-click the 4D Tools icon or select the 4D Tools icon and choose Open from the File menu.

   4D Tools displays a standard Open dialog box that allows you to select the existing database you want to work on.

2. Open the database you want to work on.

   4D Tools displays the opening screen and menu bar.

Figure 1 shows how 4D Tools looks when it's up and running.

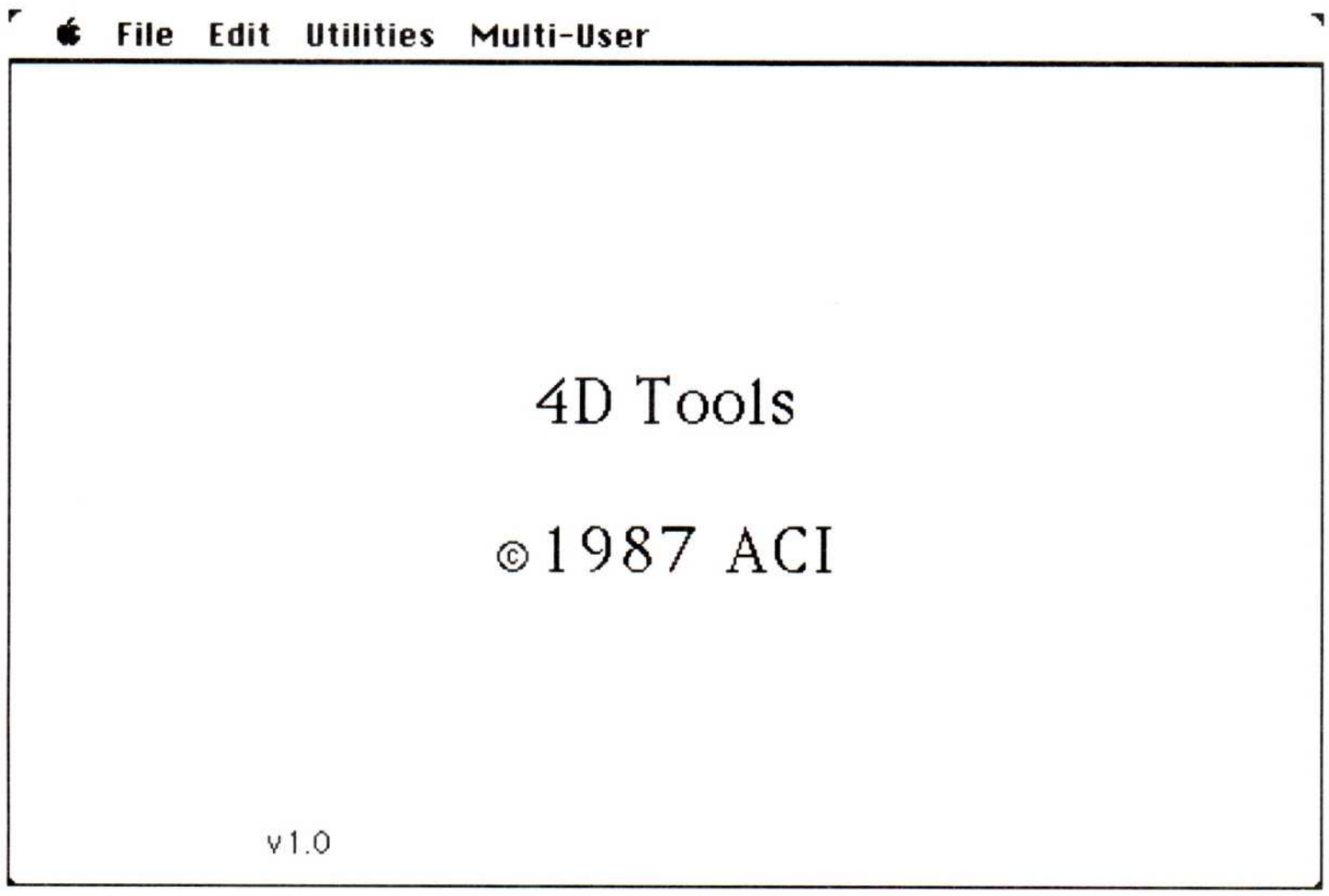

Figure 1
4D Tools opening screen

If you want to use the Multi-User menu in the 4D Tools utility, you must configure the utility for multi-user. Configuring 4D Tools for multi-user allows access to the Resize Data menu command. You should not open a multi-user database unless you are the only user.

To configure for multi-user:

1. When starting the 4D Tools utility, press the mouse button and hold it down while the utility starts.

   4D Tools displays the multi-user configuration dialog box.

2. Click Multi-user.

3. Click OK.

Warning

Do not use the utilities on a database unless you are the only user; that is, don't use them when you're working on a multi-user system. It is safest to make a copy of the database to a local hard disk, work on the copy, and then save it back to the server.

To quit 4D Tools, choose Quit from the File menu.

Cloning a database

Cloning means making a replica indistinguishable from an original. In 4th Dimension, it means making a copy of the database structure without copying any of the data that may already be in the database.

The Clone Database command copies the information created in the Design environment, all layouts, all procedures, all menus, and the database application's password system. It does not copy any data that may be in the database application itself.

Important

Some database applications provide "developer-supplied data" in one file—for example, a table of interest rates—and expect user data in other files. Export such supplied data from the old database and read it back into the appropriate new file after cloning the database.

To clone a database:

1. Start the 4D Tools utility.

   The Clone Database command will replicate the database you choose when starting 4D Tools.

2. Choose Clone Database from the Utilities menu.

   4D Tools displays a dialog box in which you can enter the name of the new database. The new database must have a different name from the old name or reside in a different folder. Figure 2 shows this dialog box.

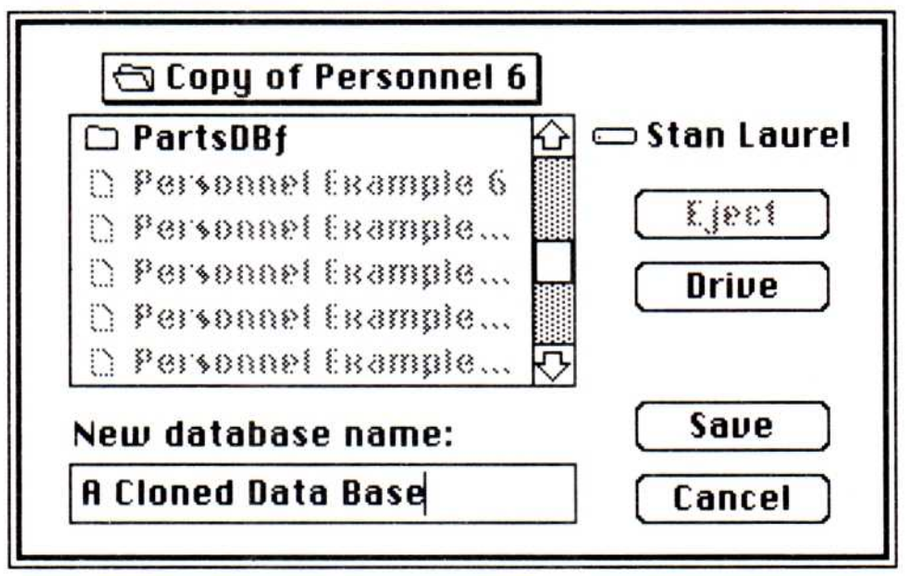

Figure 2
Naming a cloned database dialog box

3. Click Save to clone the database and give it the name you have just entered.

When it's finished cloning the database, 4D Tools remains ready for another command.

Repairing database files

The Repair Database command fixes the kind of file damage that can result from a power failure or from improper termination of 4th Dimension (for example, by turning off the power instead of quitting the program).

When you have a problem with a database, the *first* action you should take is to revert to the backup database that you maintain in parallel with your active database. Your second action should be to copy the damaged database. Do repair work *only* with the copy of the damaged database.

Repairing a database can take time. Make a backup of the damaged database *before* you begin the repair process. If something else goes wrong—say, another power failure—you still have the original, damaged database as well as a working backup database.

The Repair Database command can repair files, reindex the database, and repair links between files.

Repairing the files will remove any damaged records from the database. It will also recover any space taken by deleted records, thereby reducing the size of the database.

Repairing the indexes will rebuild all index files for the database. This can also be accomplished by turning the Index field attribute in the Design environment off and then on again.

Repairing the links will rebuild any links in the database. A rebuilt link will always be to the first matching record in the linked file. If the links were created using the second optional argument in the LOAD LINKED RECORD command, some rebuilt links may be incorrect. This happens if the user chose a linked record *other than* the first matching record.

Warning

Always back up your damaged database before using the Repair Database command.

To repair a database:

1. Start the 4D Tools utility.

   The database you select during startup is the database on which the Repair Database command will work.

2. Choose Repair Database from the Utilities menu.

   4D Tools displays the dialog box for repairing. 4D Tools will conduct each of the three repair categories that checked. Figure 3 shows the dialog box.

Figure 3
Repair Database dialog box

   Click to uncheck any of the three repair categories that you do *not* want 4D Tools to undertake.

3. Click OK to repair the database.

   When 4D Tools is finished restructuring the database and recreating indexes and links, it remains ready for your next command.

Clicking Cancel returns you immediately to 4D Tools.

Resizing multi-user data files

If you are using a multi-user database application, you must estimate the total number of records that will be saved. After creating a new database or when the number of records has grown too large for a previous allocation, use the Resize Data command in the 4D Tools utility. This command also exists in the Design environment of 4th Dimension when it is configured for multi-user.

The allocation is for record pointers, *not* data space. 4th Dimension initializes a database to hold 500 records.

Whether it operates as a single- or multi-user database, 4th Dimension allocates record pointers in blocks of 500. With a single-user database, this allocation is automatic. With a multi-user database, however, 4th Dimension cannot dynamically resize the space it allows for record pointers. Therefore, for a multi-user database application, you must estimate the number of records the database will use.

4D Tools always rounds *up* in multiples of 500 the number of records entered into the dialog box for resizing. Figure 4 shows the dialog box.

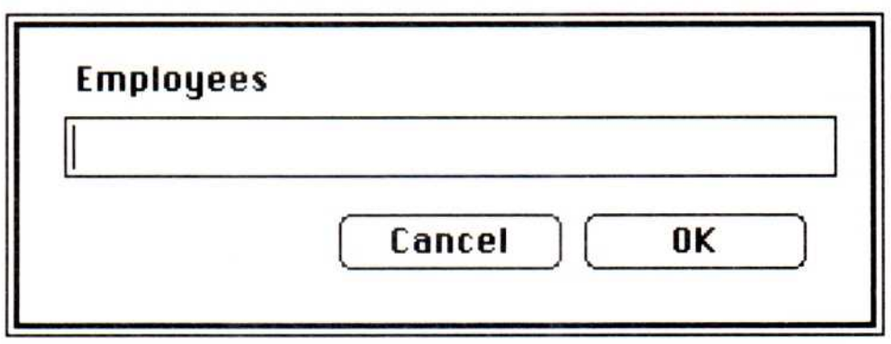

Figure 4
Resizing the files dialog box

To resize data for a multi-user database application:

1. Start the 4D Tools utility for multi-user operation.

   The database that you specify during the startup procedure is the database that the utility will resize.

2. Choose Resize Data from the Multi-User menu.

   4D Tools displays the Number of records for the file dialog box.

3. Enter the number of records you estimate you'll require for your application.

   4D Tools always rounds up in increments of 500. For example, an entry of 600 will result in an allocation of 1000 records.

4. Click OK.

5. 4D Tools will display the Number of records for the file dialog box for *each* file in the database. Repeat steps 3 and 4 for each file.

4D Renamer utility

The 4D Renamer utility simplifies the task of renaming 4th Dimension database files. It also protects you from the possible error of not properly renaming all the files in a database.

To use the 4D Renamer utility:

1. Start the Renamer by double-clicking its icon in the Finder or by selecting the Renamer icon and choosing Open from the File menu.

   The Renamer displays a dialog box showing the 4th Dimension files in the current directory. As with any Open dialog box, you can change directories; but the Renamer will show *only* 4th Dimension files.

2. Select a 4th Dimension database to rename and click OK.

   The Renamer displays its dialog box for changing the name of a database. Figure 5 shows the dialog box.

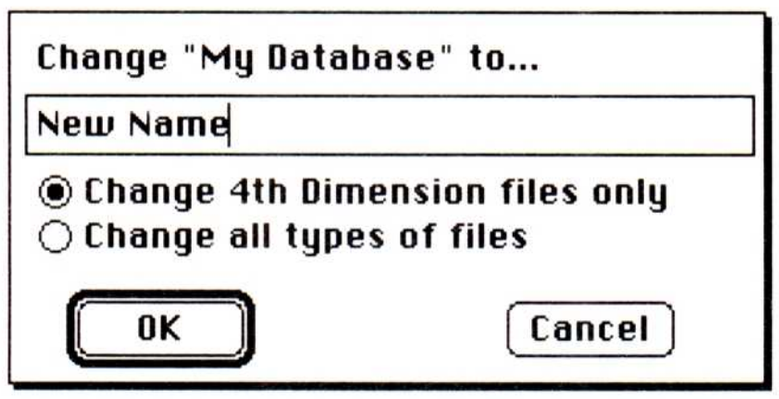

Figure 5
Changing the name dialog box

3. Enter the database's new name in the text field.

   A 4th Dimension database name cannot exceed 24 characters. The Renamer will keep you from going over this limit.

   Click Change 4th Dimension file only or Change all types of files, depending on which way you want to use the utility.

   When you click Change all types of files, the Renamer will change all files in the current directory that begin with the database's old name, whether or not they are 4th Dimension files. However, the maximum of 24 characters remains.

4. Click OK to execute the change.

   The Renamer counts the files in the directory and makes sure that the new name will not result in a filename longer than 31 characters (including index numbers and filename extensions). When the program has finished counting the files in the database, it presents the Proceed with renaming? dialog box.

5. Click OK to proceed with the renaming process.

4D Customizer utility

With the 4D Customizer, the developer can adjust attributes that affect the way the 4th Dimension program and its applications operate. The 4D Customizer groups these attributes this way:

- setting the pointer spin rate during processing
- enabling desk accessories during printing or while 4th Dimension is displaying its progress thermometers
- selecting new keys for acceptance, rejection, and entry into included layouts during input
- selecting new keys for acceptance and rejection of dialog layouts
- altering the Custom window size and type
- controlling certain aspects of the Custom window's behavior while the application is running
- setting the size of the stack and the heap

The 4D Customizer will customize the 4th Dimension program, the runtime version, or a database itself. It modifies (or creates, as necessary) the CUST resource in the program or runtime or in the .Res file of a database.

Important

The CUST resource in a .Res file of a database always overrides the program's CUST resource.

Warning

The 4D Customizer modifies the actual 4th Dimension program file or database. It is a wise policy to make a backup before altering it.

Using the File menu

Start the Customizer from the Finder.

Figure 6 shows how the Customizer looks after opening a file.

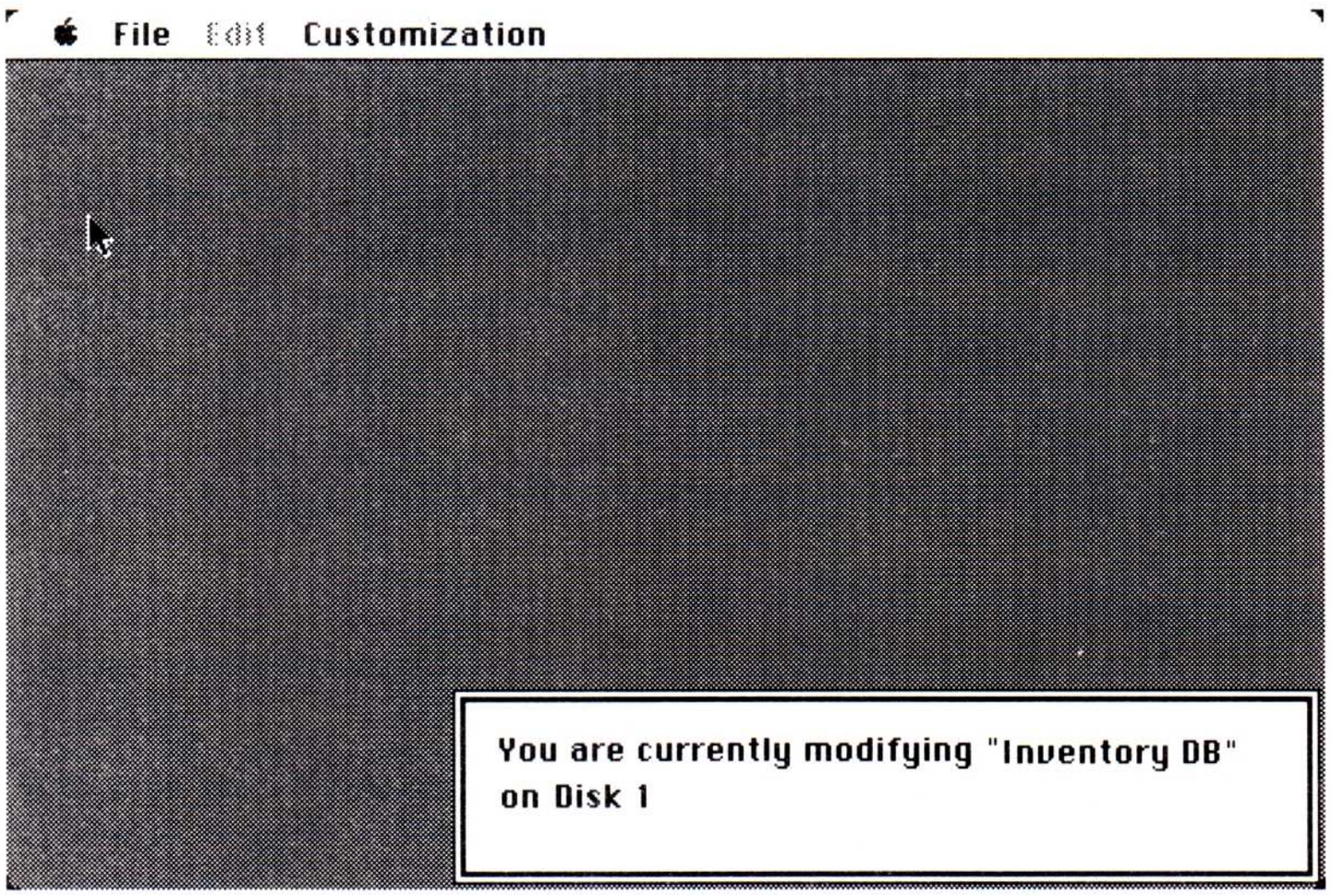

Figure 6
4D Customizer opening screen

There are six commands and a Quit command on the File menu:

- **Open Resource File** displays a standard Open dialog box. Use the Open dialog box to select the copy of 4th Dimension, the runtime version, or a database's .Res file that you want to customize.
- **Save Changes** saves the custom changes back to the 4th Dimension file you originally opened.
- **Set to Defaults** is a quick way of returning 4th Dimension to the conditions with which it was originally shipped.
- **Erase Custom Resource** removes the custom resource from the currently selected file. This command will not work on the 4th Dimension program or runtime.
- **Close Current Copy** closes the 4th Dimension file you opened first, asking if you want to save any changes you have made.
- **Transfer to Current File** asks if you want to save any changes you made, then launches the program you are modifying, bypassing the Finder. This command is selectable only if you are modifying the program or runtime.

The Quit command returns you to the Finder. If you have made any customization changes and have not saved them, 4D Customizer will give you a last-minute chance to save your changes.

To customize 4th Dimension:

1. Start the 4D Customizer.
2. Choose Open Resource File from the File menu and open the copy of 4th Dimension, the runtime version, or the database .Res file that you want.

   The Customizer continuously displays a window to remind you of which file you're modifying and its location. Figure 7 shows this window.

Figure 7
Filename and location window

3. Choose the commands you need from the Customization menu.
4. Choose Save Changes from the File menu and Quit the program.

The following sections discuss the customization changes you can make.

Setting the pointer spin rate

When 4th Dimension is executing a procedure, it normally displays the standard arrow pointer. You can change this so that it displays a rotating disk, called the beach ball, instead (see Figure 8). Using the beach-ball pointer can be useful to give you some feedback when your procedures are executing.

Figure 8
Beach-ball pointer

You can add the beach ball to your own applications and control its spin rate by choosing Pointer Spin Rate from the Customization menu. 4th Dimension displays a spin rate dialog box, shown in Figure 9.

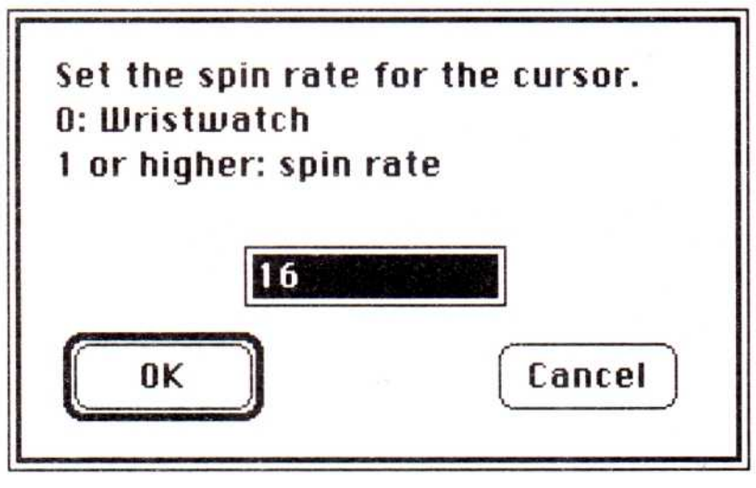

Figure 9
Setting the spin rate dialog box

Setting the spin rate to 0 tells 4th Dimension to use the standard pointer. A number greater than 0 calls for the beach ball and controls its spin.

Enabling desk accessories

As a default, 4th Dimension disables desk accessories during printing and when displaying its progress thermometers on screen. Thermometers give you a graphic idea of how much you've processed and how much there is yet to go.

You can enable the desk accessories by choosing the commands DA's Enabled During Printing and DA's Enabled During Thermometers from the Customization menu.

4D Customizer places a check mark next to each choice you make. Choose the command again to disable the desk accessories and return 4th Dimension to its default condition.

Changing input keys

4th Dimension provides keyboard equivalents for clicking OK, clicking Cancel, and for adding a new subrecord in an included layout. Table 1 shows the program's default keyboard equivalents for these tasks.

Table 1
Keyboard equivalents

Action	Equivalent
Clicking OK	Enter
Clicking Cancel	Command-. (period)
Adding a new subrecord	Command-Tab

To change the keyboard equivalents of these three actions:

1. Choose the appropriate command from the Customization menu: Input Layout Acceptance Key, Input Layout Rejection Key, or Input Layout New Subrecord Key.

   4th Dimension displays a dialog box in which to enter an ASCII code for the key, and three check boxes representing the potential modifiers of the key: Shift, Option, or Command.

2. Enter the ASCII code of the key you want as the keyboard equivalent.

Important

If you enter an ASCII code that is generated with the use of a modifier key such as Shift or Option, make sure to check the modifier box also.

---

3. Click the check boxes for the modifiers, if any.

   For example, if you want the keyboard combination of Option-Tab to signal acceptance of a record (equivalent to clicking an OK button), you would enter the ASCII code 9 in the box and click the Option check box.

4. Click OK or Cancel.

Important

Be wary of making the Return key the equivalent of clicking OK. If you do, you will not be able to use returns in Text fields or the Return key to move from field to field within a layout.

Figure 10 shows the dialog box that allows you to change the keys.

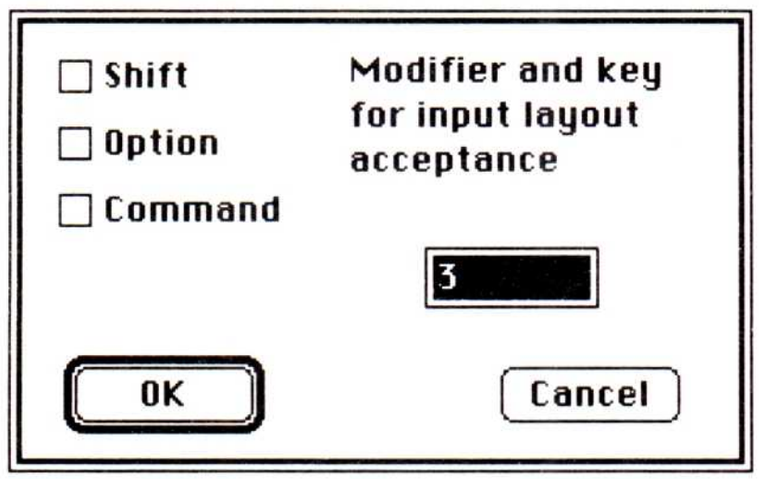

Figure 10
Changing the keys dialog box

Changing dialog box keys

You can change the acceptance and rejection keyboard equivalents for a dialog layout by choosing the commands Dialog Acceptance Key and Dialog Rejection Key.

4th Dimension's defaults for acceptance and rejection of a dialog layout are the same as for an input layout: Enter and Command-. (period). 4D Customizer displays a similar dialog box for changing the keyboard equivalents as it does for the input keys.

Setting the window

You can control how the input window appears in the Custom environment with these options:

- **Standard Window** produces a full-size window displaying the application name in the title bar. This is the default choice.
- **Standard Window, But No Title** produces a full-size window with no title bar.
- **Custom Window Size** displays a dialog box where you can set window margins and center the window in the screen.

When you select one of the three commands, 4D Customizer places a check mark next to it. If you select Custom window, 4D Customizer displays a dialog box where you can set the dimensions of the window and center it. Figure 11 shows the Custom window dialog box.

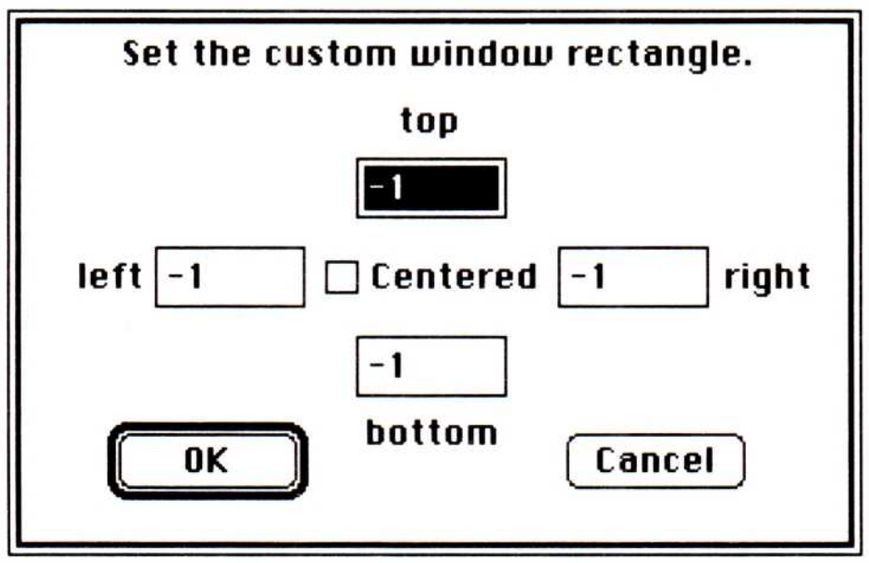

Figure 11
Setting the Custom window dialog box

Important

The –1 values in all four Custom window margin boxes are 4D Customizer's default values, but do not in themselves produce a standard-size window in 4th Dimension.

Setting window controls

The next two choices in the Customization menu affect how you the developer want 4th Dimension to control the input window for your users. When you select either or both of the window control setting choices, 4D Customizer will place a check mark next to the choice.

- **Keep User Environment Window Size in Custom:** Whatever window size is set by the user in the User environment continues to hold in the Custom environment. The default is for 4th Dimension to use the window size and type set by the previous three choices.
- **Revert to Original Window Size After Input:** Some users like to control the size of the window during the input process. In either the Custom environment or the User environment, setting this command means that whenever 4th Dimension displays an input layout, it displays it at the default size—and does not continue to display it the way the user may have previously set it.

Setting stack and heap size

The final two settings control the way 4th Dimension and its database applications interact with the memory of the computer. To reach them, drag below the bottom of the open Customization menu; the menu choices will scroll.

Warning

Changing the size of the stack and the heap can directly—and dramatically—affect how 4th Dimension and applications work. Refer to *Inside Macintosh* for a thorough discussion of the stack and the heap **before** changing their values for your program.

The stack is where the computer stores the values of 4th Dimension's own variables, subroutine calls, intermediate mathematical results, and so forth. The stack grows and shrinks as 4th Dimension operates. Setting the stack limits the amount of memory that the computer reserves for 4th Dimension (*limit* can also mean "expand beyond the default amount").

The heap is the area of memory left over after the computer has assigned memory to programs and their data. It holds the content of variables from 4th Dimension's database application and developer programming, all resources, and handles.

By *decreasing* the size of the heap, you can make it possible for an application to run on a smaller machine—for example, a Macintosh 512K Enhanced with a large disk cache. Because of the heap's limited size, the application will also run more slowly.

By *increasing* the size of the heap, you may be able to speed up a database application; but you may also make it necessary to run it on larger machines.

A setting of 0 for either the stack or the heap tells 4th Dimension to operate at its defaults.

To set the stack size:

1. Choose Set Stack Size from the Customization menu.

   4D Customizer displays its dialog box for setting the stack size. The box shows the currently set figure for the stack in bytes. If you used the File command to return the 4th Dimension program to its defaults, the box will show 0.

2. Enter the maximum size of the stack you want to set.

   Figure 12 shows the dialog box for setting the stack size. An acceptable size for the stack is between 20,000 and 256,000 bytes (or 0). If you enter a figure out of this range, 4D Customizer will warn you.

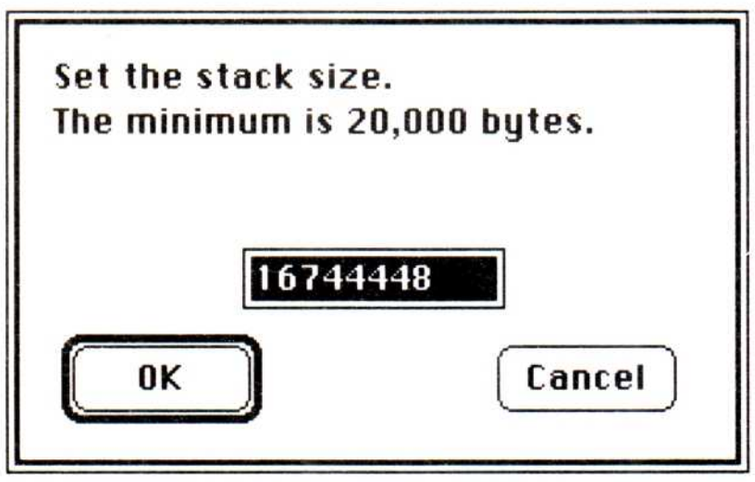

Figure 12
Setting the stack size dialog box

3. Click OK or Cancel.

To set the heap size:

1. Choose Set Heap Size from the Customization menu.

   4D Customizer displays its dialog box for setting the heap size, which is similar to the dialog box for setting the stack size.

2. Enter the minimum heap size you wish to set.

   A safe minimum heap size is 200,000 bytes. You may also enter a 0 to tell 4th Dimension to use its defaults.

3. Click OK or Cancel.

4D Mover utility

4D Mover converts 68000 object code to 4th Dimension external procedures and transfers them between files much the way that the Font/DA Mover transfers fonts and desk accessories. An external procedure is a program written outside of 4th Dimension in any high-level language such as Pascal, C, or 68000 Assembly and compiled and linked or assembled to 6800 object code. Using 4D Mover, you can add that procedure to 4th Dimension or your application.

Warning

Before you use 4D Mover, you should read Appendix D, on external procedures, in *4th Dimension Command Reference*. 4th Dimension hands over *complete* control to an external procedure.

4D Mover works with five types of files:

- **4th Dimension program:** You can add your own external procedures to your copy of the 4th Dimension program. There, they are available to any application running under the modified copy of the program.
- **4th Dimension runtime version:** You can add external procedures to the runtime version, as you can to the 4th Dimension program.
- **4th Dimension database application .Res file:** Adding external procedures to the database application .Res file makes them available only to that database application, but available when running under *any* copy of 4th Dimension or the runtime version.
- **68000 object code files:** These are compiled and linked application files that contain the code that 4D Mover converts into 4th Dimension external procedures.
- **External procedure Library files:** A Library file acts as a "holder" for one or more external procedures—much like a Font/DA Mover "suitcase" file. 4D Mover creates Library files.

A library file named Proc.ext can exist in a database folder or in the system folder of the startup disk. Adding external procedures to a file named Proc.ext file in a database folder makes them available to that database. Adding them to a file called Proc.ext in a system folder makes them available to all databases run under that system.

Preparing

To be able to move an external procedure into a file, you must first create the procedure in the form of an application. Appendix D in *4th Dimension Command Reference* contains complete guidelines for creating such a procedure. In summary, they are

- 4th Dimension hands over complete control to the external procedure. The procedure must "know" what to do, and "clean up" before it hands control back to 4th Dimension.
- The entire code of the application you intend to convert (the object code) must be only one segment of less than 32K (32,767 bytes).
- All parameters in your procedure must be *variable* parameters. You may include up to 100 parameters in your procedure. Parameters in your procedure and as described to 4D Mover must be the same in type, order, and number.
- 4D Mover copies the CODE segment with an ID of 1 from your compiled and linked application, and pastes it into a resource called 4DEX that belongs to the file in which you intend to put the external procedure. If you have compiled and linked your procedure as an application, then CODE begins with the 4 bytes of the segment jump table. 4D Mover removes these 4 bytes before pasting the code in 4DEX. If you have only compiled your procedure or have instructed the linker to not generate the segment jump offset, then 4D Mover allows the option of not removing the first 4 bytes.

Starting

Start 4D Mover from the Finder. Figure 13 shows how 4D Mover looks when it starts.

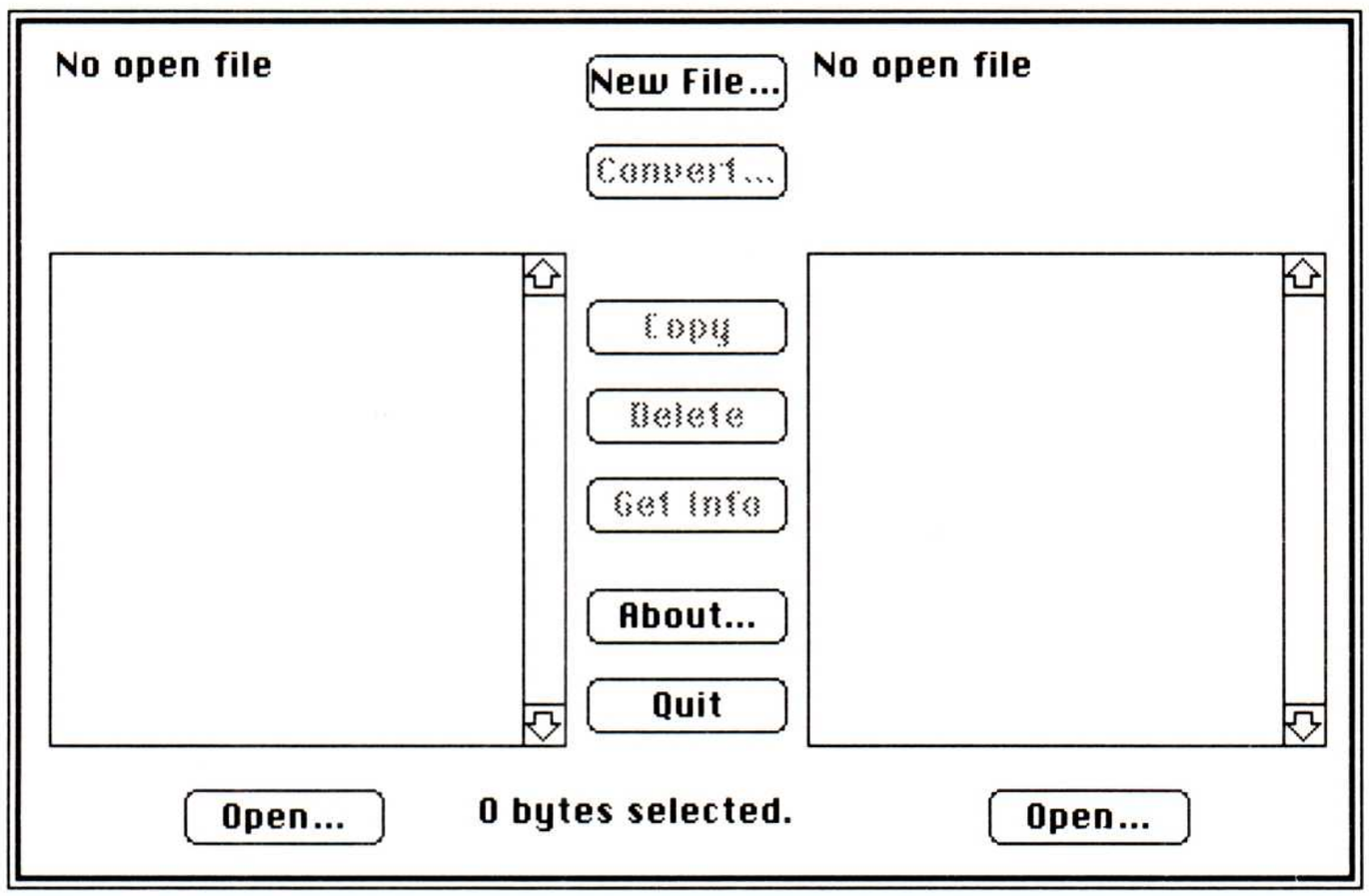

Figure 13
4D Mover opening screen

As with Font/DA Mover, you may click Open on either side of 4D Mover first.

Creating a Library file

Library files hold one or more procedures *that have already been converted* for use with 4th Dimension. They are like the "suitcase" files that hold DA's and Fonts. New Library files are, of course, empty.

To create a new Library file:

1. Start 4D Mover.
2. Click New File.

4D Mover displays a dialog box in which you can name the new Library file. Figure 14 shows this dialog box. 4D Mover proposes the name **external proc's** as a default Library filename.

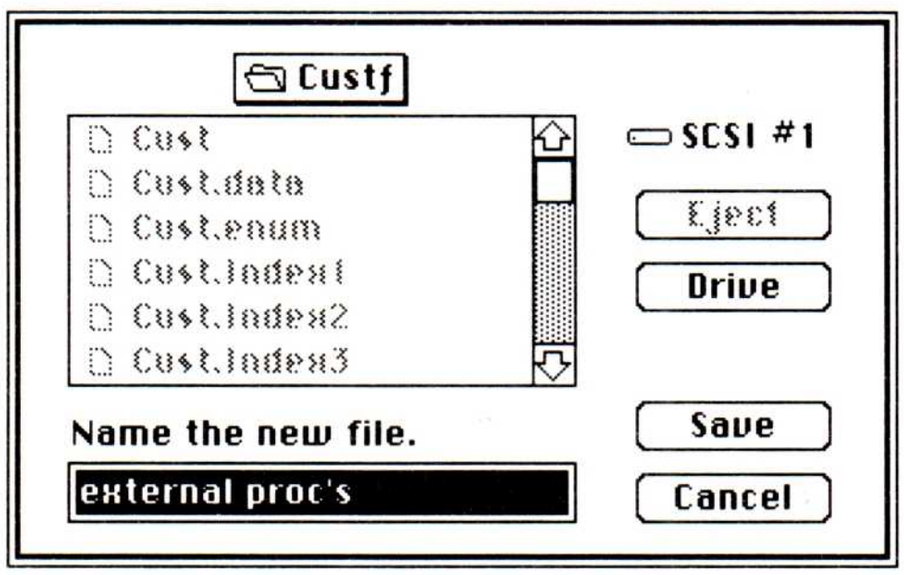

Figure 14
Naming the new file dialog box

3. Click Save.

   This creates the new Library file and opens it in one side of 4D Mover's main screen.

You may now copy external procedures into the Library file.

Converting an application to an external procedure

After compiling and linking your source code into an application, you can convert it into an external procedure using 4D Mover. Converting an application is a different action from adding the external procedure to 4th Dimension, its runtime, an application, or a Library file.

Important

4D Mover does not cannibalize the original application code. You can safely convert any appropriate application; 4D Mover will leave the original code intact.

To convert an application to an external procedure:

1. Start 4D Mover and click Open (either side).

   4D Mover displays a standard Open dialog box. The dialog box recognizes only .Res files, 4th Dimension program files, runtime files, and Library files.

2. Open the *destination file* for your conversion.

   Often, this will be a Library file, but it may be the 4th Dimension, runtime, or database application. It's generally easier and faster to convert several applications into external procedures and house them in a Library file than to convert them directly to the 4th Dimension program.

4D Mover will list the number of external procedures in the file. 4D Mover will delete any Library file containing 0 external procedures when you close the Mover, but will not delete 4th Dimension, runtime, or database application files containing 0 external procedures.

3. Click Convert. 4D Mover displays a standard Open dialog box.

   If you hold down the Option key while you click Convert, 4D Mover displays a dialog box that asks you if you want to remove the 4 bytes from object code of your application. If your linker can do this (with MPW you can, for example), you may link your application without the 4 bytes.

   Your choice in this dialog box becomes the Mover's default. If more than one person uses 4D Mover regularly, you may want to make it a practice to check this default yourself.

4. Open the application you want to convert.

   If both sides are open, 4D Mover asks to which side you want the conversion made. Figure 15 shows this dialog box.

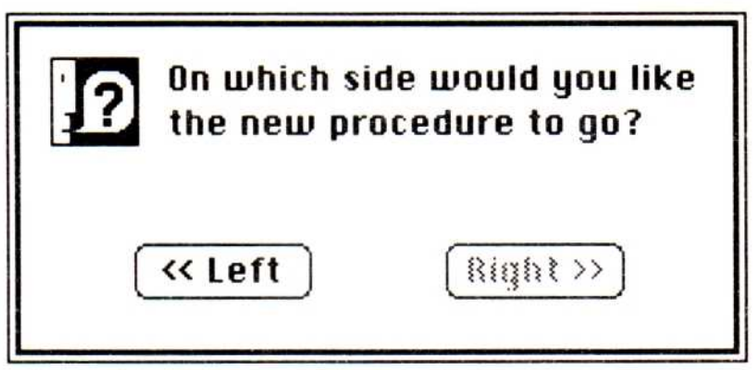

Figure 15
Choosing left or right side dialog box

5. Click the side to which you want the conversion moved.

   4D Mover displays a screen on which you can enter the parameters of your procedure (and their order), add comments, and enter the procedure name. Figure 16 shows this screen.

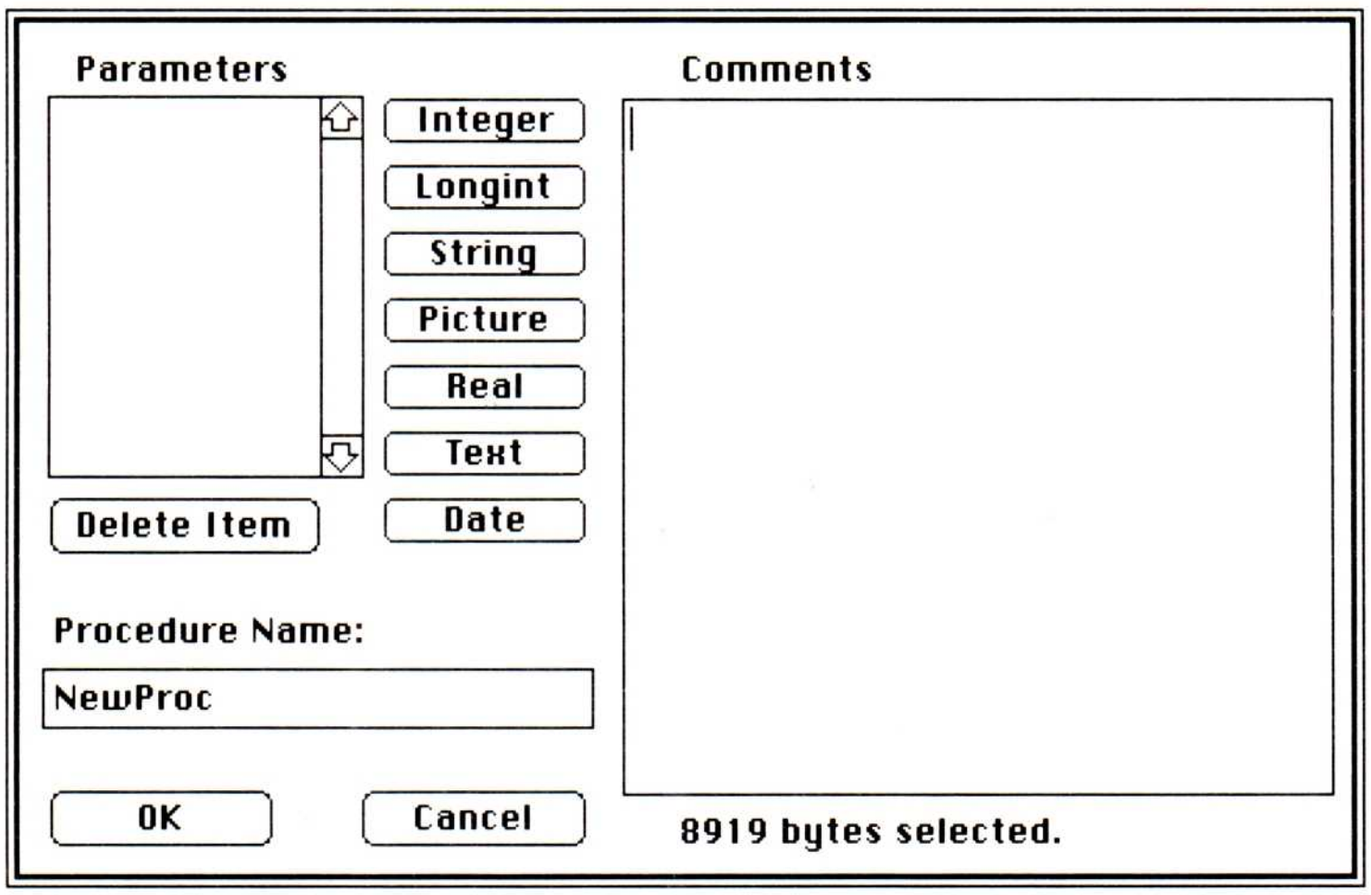

Figure 16
Entering parameters, comments, and procedure name

6. When you have finished entering parameters, comments, and the procedure name, click OK or Cancel.

Entering parameters

You *must* enter the proper number and type of parameters into 4D Mover that you have in the declaration block of your procedure. The parameters must also be in the same order as they are in your procedure source code. 4D Mover does not need to know the parameter names—just their number, type, and order.

To create a list of parameters matching that in your procedure:

1. Click a parameter type from the list of seven buttons to the right of the parameter list box.
2. Click a parameter in the list to select it. Click below the existing parameters, but inside the list box, to select no parameter.

4D mover always adds a new parameter *above* any selected parameter. If there is no selected parameter, 4D Mover adds the new parameter to the end of the list.

To delete a parameter, select it and click Delete Item.

You may enter from 0 to 100 parameters.

Entering comments

You can enter information about a procedure in the Comments area. 4D Mover will display these comments (and the rest of the parameter and procedure name information) when you click the Get Info button on 4D Mover's main screen.

To enter comments:

1. Begin typing.

   4D Mover will enter the characters you type in the Comments area. Text will wrap automatically.

2. Press Return for a combined line feed and carriage return; the Enter key is the equivalent of clicking OK.

While 4D Mover displays no Edit menu, the keyboard-equivalent editing functions Cut, Copy, and Paste (Command-X, -C, and -V) still work. Text in the Comments area will scroll up and down when the text insertion point (or selected text) exceeds the boundaries of the area.

Entering a procedure name

The name for the external procedure that you enter into the Procedure Name area need not match the name that the procedure had in its source code form. However, the name you give it here is the name you must use when writing procedures in 4th Dimension.

The name of an external procedure may be up to 15 characters long. This is the same restriction as for procedure names within 4th Dimension.

Copying an external procedure

Once you have converted an application to an external procedure, you may transfer it from file to file and from one copy of 4th Dimension to another. Copying an existing external procedure from a Library file (for example) to a copy of 4th Dimension is the same as installing it in that program.

To copy an external procedure:

1. Start 4D Mover and open a file containing at least one external procedure. You may double-click a Library file to both start 4D Mover and open the file.
2. On the other side of 4D Mover's main screen, open a destination file.

   This may be 4th Dimension, a runtime, a database application, or a Library file.

3. Select which external procedures you wish to copy from the source file.

   The Copy button activates. Angle brackets point which way the copy will proceed.

   Click to select an individual procedure, Command-click to select noncontiguous procedures, and Shift-click to select contiguous procedures.

4. Click Copy.

4D Mover copies the procedure from one location to the other.

Deleting an external procedure

To delete external procedures from a file, open 4D Mover and the file, select the procedures you wish to delete, and click Delete.

4D Mover will confirm that you do, indeed, wish to delete the selected items. Click Yes to continue, No to stop.

Important

4D Mover automatically deletes any empty Library files.

Using other command buttons

The Get Info button on 4D Mover's main screen displays the parameters, comments, and procedure name of the selected procedure. You may use Get Info for only one external procedure at a time.

The About button displays the author and copyright statement.

Summary of command buttons

This list summarizes the command buttons from the main screen of 4D Mover and what they do:

- **New File** creates an empty Library file for external procedures.
- **Convert** opens the application file you want to convert to an external procedure and displays the 4D Mover screen showing parameters, comments, and procedure name.
- **Copy** transfers an existing external procedure from one file to another. When you transfer an external procedure to 4th Dimension, its runtime, or a database application, it is equivalent to installing it.
- **Delete** deletes the selected external procedure from its file.

- **Get Info** displays the 4D Mover screen showing parameters, comments, and procedure name for the selected procedure. You may change any or all of the three items.
- **About** displays the author and copyright statement.
- **Quit** exits 4D Mover.
- **Open/Close** opens and closes source and destination files that hold external procedures.

Multi-user applications

Every copy of 4th Dimension and the runtime version inherently supports multi-user applications. No extra programming is necessary to run a 4th Dimension application in a multi-user environment. However, some special attention is advisable to obtain the best performance, security, and data integrity from a 4th Dimension application running in a multi-user environment. You may operate a multi-user 4th Dimension application in either the Custom or the User environment.

Under certain conditions, a user may enter the Design environment even when using 4th Dimension in multi-user. See "Switching Between Environments" later in this section.

Multi-user functionality permits simultaneous access to a database by more than one user. Users can perform different operations on the same database at the same time, such as data entry or retrieval, record modification, manipulation, display, or printing.

Record locking prevents collisions between users. If one user is modifying a record, then no other user can modify the same record. 4th Dimension allows other users to see the record and use the data in any way except to modify it.

4th Dimension is compatible with file servers having a full implementation of the Hierarchical Filing System (HFS), for example AppleShare™ by Apple Computer, Inc..

The 4th Dimension database application resides on the server. Each individual user must have a copy of 4th Dimension or the runtime version running on his Macintosh. 4th Dimension coordinates the data access by different users and avoids collisions.

The Resize Data command

When using a database with multi-user, you need to predeclare the maximum number of records the database will have. New databases are always created with the default number of records set to 500. To resize the database, use the Resize Data menu command in 4D Tools or in the Design environment of multi-user 4th Dimension.

When a multi-user database approaches the allocated number of records (within 5, for example on the 495th record), the database needs to be resized. The size is always in multiples of 500; so if you entered 750, it would resize for 1000 records.

For more information, see "Resizing Multi-User Data Files" in the section "4D Tools Utility" in this supplement.

Starting

After you have developed a multi-user application, copy the application to the file server and set the access privileges on the file server for all the users who will be running the database application.

You must have a 4th Dimension program or runtime version for each user.

To bring up the application:

1. Start 4th Dimension (or runtime) on any Macintosh that is connected to the file server. While the program starts, *hold the mouse button pressed.*

   4th Dimension displays a dialog box in which you can set the option of using the program as single-user or multi-user. Figure 17 shows this dialog box.

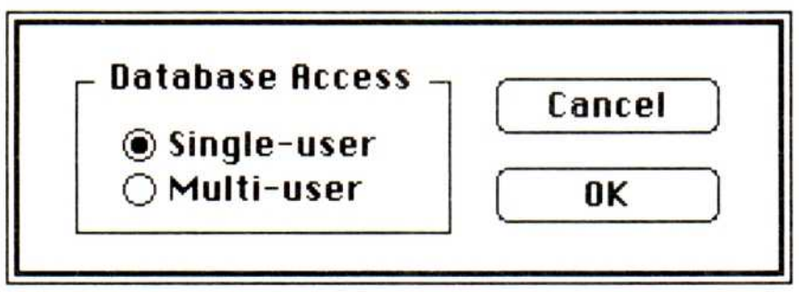

Figure 17
Setting single- or multi-user dialog box

2. Click Multi-user and OK.

   4th Dimension uses this setting as its default. Each time a user starts this copy of 4th Dimension, it opens as a multi-user system. To change back, repeat step 1 and click Single-user.

   4th Dimension displays its Open database dialog box.

3. Select the correct drive (the file server) and open the database.

Each user who is going to access the database over the network must set up his copy of 4th Dimension as a multi-user system as in steps 1 and 2.

Important

If a user starts 4th Dimension as a single-user system and accesses the database over the network, no other users may log onto the database. An error message is displayed.

Switching between environments

A user of a 4th Dimension multi-user database application can switch only between the User and the Custom environments.

However, if there is only one person on-line, he is permitted to switch to the Design environment. While he is there, no other user may open the database. If another user attempts to log on, the system displays an error message stating "This database structure is being modified."

When the user in the Design environment switches back to the User or Custom environment, other users may access the database once again.

Understanding the features and functions

4th Dimension has complete multi-user functionality built in. Any 4th Dimension database application will run *as it is* as a multi-user application. Multi-user support extends to both the Custom and the User environments.

4th Dimension allows this automatic multi-user functionality by setting a default read-write flag for all commands *except* those that involve displaying or printing. Thus, the progam automatically locks against modification by all other users the record any one user is modifying. Those other users may obtain a read-only copy of the record in use, and can use any of the data for operations such as printing reports. Users who have a read-only copy of a record are not able to save any changes to that record.

Display or print operations require a read-only copy of the record; there is no need to lock the record against other users. Therefore, 4th Dimension changes the state to read-only. When one user is printing a record, another can modify it and save the changes.

When a user has access to a read-write file and executes a command that selects a record of the file, 4th Dimension checks to see if the record is locked.

If the record is free, 4th Dimension locks it against change by any other user and loads that record as the current record for the user. If the record is already locked, 4th Dimension still loads it as the current record for the user. If the selection command was **MODIFY RECORD**, or the user double clicked to modify a record in a **MODIFY SELECTION** listing, then a warning "This record is being modified" is displayed. 4th Dimension does not allow any modifications to the record to be saved by other than one user.

Each user has his own current record and current selection for every file in the database. A user cannot save the current record of any file unless that file is in a read-write state and the record is not locked.

Accessing a record by one user in a read-write state locks only the record, and not the entire file, for other users. All global file operations such as search or sort affect only the user's current selection and not the database resident on the file server.

When a record is in use and locked to others, it remains locked until the user changes his current record. Thus, when one user saves a record, it is still the current record and remains locked to all other users until is unloaded.

Printing and displaying always set the state to read-only. After display or printing is over, 4th Dimension resets the record to its previous state (read-write or read-only).

Programming

4th Dimension provides an extensive set of procedures and functions for programming complex multi-user applications. Typically, you would program a database application specifically for multi-user operation if you wanted to change the default from read-write to read-only, if you required unique numbers applied once to individual records, or if you wished to batch-process several records.

For example, when a user is trying to access a locked record, you may want to give that user a choice of waiting for the record to be free, canceling the operation, or taking some other action.

4th Dimension handles multi-user operations by locking the current record of any file against all other users. If a user searches for a record, modifies it, and saves the changes, at that point—although the user is finished working with the record—the record remains loaded and locked against other users. To unlock the record, the user must do another operation that changes the current record, or the procedure must explicitly call **UNLOAD RECORD**. Otherwise, the record remains locked.

Special circumstances

There are circumstances that require special attention to multi-user programming. For example, consider an invoicing system where each invoice must have a unique invoice number.

To assign the number, you could use a variable that the numbering procedure reads in and increments during the Before phase of the execution cycle. However, if the user then fails to save the invoice, the application would skip the number.

The best way to handle this typical situation is to assign the invoice number in the After phase of the execution cycle, so that the procedure will assign a number only if the user saves the invoice. Here again, two users may both read the "last number" saved to disk and simultaneously increment to the next proper invoice number, causing duplicates.

To make sure that the numbering procedure does not duplicate invoice numbers, use 4th Dimension's Semaphore command. The assigning procedure of each user tests if Semaphore is set (is True). If it is, the procedure must wait. If not, then it creates the semaphore, thus setting Semaphore to True, loads the "last number," increments it, and saves the record and the new "last number" to disk. Then, the procedure issues CLEAR SEMAPHORE. In this way, while the semaphore is set, only one user can do the saving. This will keep the invoice numbers unique.

You can lock multiple records as well. For example, if you are creating an invoice that includes several items from inventory, you may want to lock all the items you are using so that there is no change in quantity other than the one record you are creating.

To do this operation, you must be in the read-write state and use PUSH RECORD for all the items you want to lock against others. PUSH RECORD pushes the current record onto the file stack along with its state (read-write or read-only).

The Flags file

The *database*.flags file contains semaphores for each active user of a multi-user database application. If a user leaves the application improperly, for example by turning off the power instead of quitting, then the Flags file will not be updated properly. You might notice this if 4th Dimension said you were User #2 when in fact you were the only user. To fix this problem, simply wait until there are no users and then throw away the Flags file.

Commands

The following pages describe commands specific to multi-user operation. The command syntax is given, followed by a description, an example, and references to related commands described in *4th Dimension Command Reference*.

CLEAR SEMAPHORE

Syntax CLEAR SEMAPHORE (*strexpr*)

Description CLEAR SEMAPHORE erases the flag *strexpr* created by the Semaphore function. 4th Dimension creates semaphores on the file server in a file called *database*.flags.

There are some semaphores that the system uses. You must not clear the system semaphores SETUP, INDEX, STRU, and the numbers 1 through 64 created for each user.

Example
```
CLEAR SEMAPHORE ("Beowulf")
```

References LOAD RECORD, Locked, READ ONLY, READ WRITE, Semaphore.

LOAD RECORD

Syntax LOAD RECORD« (*filename*) »

Description LOAD RECORD loads the current record of *filename* into memory. Multi-user applications use LOAD RECORD to test whether or not a record is locked. It is not necessary in single-user applications because 4th Dimension automatically loads the current record of a file. In multi-user applications, a record may be in memory, but locked because it is in use. You may then need to load the record, which will reload it and update the status of locked or free.

If you don't specify *filename,* LOAD RECORD uses the default file. If no current record exists, LOAD RECORD has no effect.

After trying to load a record, you can test Locked to see if that record is in the read-write state for any other user.

Example
```
While (Locked ([Customers]))
   LOAD RECORD ([Customers])
End while
```

References Locked, READ ONLY, READ WRITE, UNLOAD RECORD.

Locked

Syntax

Locked« (*filename*) »

Description

Locked is a function that returns a Boolean value. If the current record of *filename* is in read-write state for any one user, then it is locked to all other users and Locked will return True for all the other users.

If you don't specify *filename,* Locked uses the default file.

If a record is locked, then the user can modify it, but cannot save the changes. Use this function to find out if the record is locked; then take appropriate action—such as giving the user a choice of waiting for the record to be free, skipping the operation, or taking some other action.

Example

```
SEARCH
While (Locked ([Customers]) & (OK=1))
   Confirm ("This record is being used by someone else. Do you want to wait for it
         to be free?")
   LOAD RECORD ([Customers])
End while
```

References

LOAD RECORD, READ ONLY, READ WRITE, UNLOAD RECORD.

READ ONLY

Syntax

READ ONLY« (*filename*) »

Description

READ ONLY changes the default status of any record of *filename* that is loaded in memory to read-only. All subsequent records that are loaded will be in the read-only state, and the user will not be able to save any changes made to them.

If you don't specify *filename,* READ ONLY uses the default file.

Use this command when the user is doing something that does not require the loaded record to be locked unnecessarily for all other users. If a user is doing anything that requires the record *as it was last saved,* and will not change any data in the record, then the user should have a read-only copy of the record. Another user may want to change it in the meantime.

4th Dimension sets the state automatically to read-only whenever it performs any display or print command. After such a command, 4th Dimension resets the state of *filename* to its previous state (read-only or read-write).

Example

READ ONLY ([Customers])

References

ADD RECORD, CREATE RECORD, LOAD RECORD, Locked, MODIFY RECORD, READ WRITE, SAVE RECORD, UNLOAD RECORD.

READ WRITE

Syntax

READ WRITE« (*filename*) »

Description

READ WRITE changes the default status to read-write for any record of *filename* that is loaded into memory. All subsequent records that are loaded will be in the read-write state if their records are not specifically locked, and will remain locked to all other users.

If you don't specify *filename,* READ WRITE uses the default file.

Use READ WRITE when the user must modify the record and save the changes he makes.

The default in 4th Dimension is read-write, with the exception of display and print commands. If you use the commands ADD RECORD, LOAD LINKED RECORD, LOAD RECORD, MODIFY RECORD, and so forth, the state must be read-write.

Use READ WRITE also when you must lock a record for other users, even if you are not making any changes such as using the total budget to determine the department's budget.

Example

READ WRITE ([Customers])

References

ADD RECORD, CREATE RECORD, LOAD RECORD, Locked, MODIFY RECORD, READ ONLY, SAVE RECORD, UNLOAD RECORD.

Semaphore

Syntax Semaphore (*strexpr*)

Description Semaphore is a Boolean function that tests to see if a flag with the name *strexpr* exists on the file server. Only one user at a time can create a semaphore. If the flag exists, Semaphore returns True. If the flag does not exist, Semaphore creates the flag and returns False.

This function is a discrete operation. If the semaphore does not exist, the function creates it. Otherwise, the semaphore is not changed by the function and the result is True. Semaphore returns False for only one user at a time.

Example

```
` In an Input Layout Procedure
Case of
   : (Before)
      ` The Before phase
   : (During)
      ` The During phase
   : (After)
     While (Semaphore ("InvoiceNo"))
        ` Wait for the semaphore to be free
     End while
        ` Assign a unique incremented number to the Invoice
     LOAD VARIABLE ("InvoiceVar"; SavedNum)
     [Invoice]Invoice Number := SavedNum + 1
        ` Save the new incremented invoice number
     SavedNum := SavedNum + 1
     SAVE VARIABLE ("InvoiceVar"; SavedNum)
        ` Clear the semaphore so that other users may save their invoices
        CLEAR SEMAPHORE ("InvoiceNo")
End case
```

UNLOAD RECORD

Syntax

UNLOAD RECORD« (*filename*) »

Description

UNLOAD RECORD unlocks the current record of *filename* for other users. When you load an unlocked record with READ WRITE set, 4th Dimension locks it to other users. Use UNLOAD RECORD when you have finished using the record.

If you don't specify *filename,* UNLOAD RECORD uses the default file.

4th Dimension automatically calls this function whenever another record is made the current record. Thus, if you are modifying all the records of a file one by one, after you save one record and call NEXT RECORD, 4th Dimension automatically unloads the previous record and makes the next record the current record.

At the end of the loop, however, when you reach the last record of the file, the current record cannot change to the "next record." You must call UNLOAD RECORD to free the last record for other users.

UNLOAD RECORD unloads the record from memory, but the record remains the current record.

Example

```
` A sample multi-user procedure to allow user to modify a record
DEFAULT FILE([Customers])
  ` Allow user to search a record
SEARCH
If (OK=1)
     ` If user confirms his search
  While (Locked & (OK = 1))
       ` While the record is locked and user wants to continue
    CONFIRM ("This record is already in use, do you want to wait?")
      ` Try to load unlocked the record by calling LOAD RECORD
    LOAD RECORD
      ` LOAD RECORD will update the status of Locked
  End while
  If (OK = 1)
       ` If we quit the loop with OK=1 Locked is now false, so the user's
       ` modifications can be saved
    MODIFY RECORD
      ` Free the record for other users on the network by calling UNLOAD RECORD
    UNLOAD RECORD
  End if
End if
```

References

LOAD RECORD, Locked, MODIFY RECORD.

Related Commands: PUSH RECORD and POP RECORD

Syntax

PUSH RECORD« (*filename*) »
POP RECORD« (*filename*) »

Description

Use PUSH RECORD and POP RECORD to lock multiple records for users. When you use PUSH RECORD, 4th Dimension pushes the current record onto the file stack and preserves the state (read-only or read-write). Thus, if the state of the record is read-write, after it is pushed, the state continues to be read-write and locked to other users. You can now change the current record, and the record you have pushed remains locked to other users.

When you use POP RECORD, the state remains unchanged. The record you pop becomes the current record, though you need to make it the current selection by using ONE RECORD SELECT. You can then unload the current record and unlock it for others.

ACIUS